实用第一 智慧密集 Practicability First, Intelligence Intensive

电脑编程技巧与维护

COMPUTER PROGRAMMING SKILLS & MAINTENANCE

http://www.comprg.com.cn

上 5月 2013年5上03日

每期定价:11.00元 全年定价:264.00元
《电脑编程技巧与维护》杂志社出版
刊号: ISSN 1006-4052 / CN11-3411/TP
广告许可证 京海工商广字0151号

国家级科技期刊 中国学术期刊综合评价数据库统计源期刊 中国核心期刊（遴选）数据库收录期刊

ISSN 1006-4052

创新驱动 应用引领
服务经济社会发展

展览日期：5月30日–6月1日

官方微博　官方网站　微信

www.csia.org.cn / www.csiaexpo.com

2013

第十七届中国国际软件博览会

INT'L SOFT CHINA

主办单位：

工业和信息化部
国家发展和改革委员会
科学技术部
国家外国专家局
北京市人民政府

支持单位：

各省、自治区、直辖市、计划单列市、
副省级省会城市软件和信息技术服务业主管部门

承办单位：

中国软件行业协会
中国电子信息产业发展研究院
中国国际贸易中心股份有限公司

协办单位：

各地方软件行业协会
工业和信息化部电子科学技术情报研究

2013第十七届中国国际软件博览会

*姓名（用于入场登记核对）：　　所在城市：

*手机号（用于发二维码）：　　QQ/微信号：

*邮箱（用于发二维码）：

单位：　　部门及职务：

通信地址及邮编：

带*号请务必填写　　参观时间：5月30日 9:30-16:00 / 5月31日 9:00-16:00 / 6月1日 9:00-13:30 最前半小时停止入场

官方微博　官方网站　微信

5月30日–6月1日

中国·北京展览馆

参观券
Admission Ticket

2013 年第 09 期
5 月(上)

电脑编程技巧与维护

总第 279 期　　1994 年 7 月创刊　　(半月刊)

编 辑 出 版：《电脑编程技巧与维护》杂志社
主 管 部 门：中华人民共和国工业和信息化部
主 办 单 位：中国信息产业商会
地　　址：北京市海淀区长春桥路 5 号
6 号楼 1209 室
投 稿 信 箱：gaojian@comprg.com.cn
gaojian@comprg.sina.net
编辑部信箱：gaojian@comprg.com.cn
发行部信箱：zzsfx@vip.sina.com
网　　址：http://www.comprg.com.cn
邮　　编：100089
电　　话：(010) 82561037
传　　真：(010) 82561614
照　　排：《电脑编程技巧与维护》
杂志社电脑排版部
印 刷 厂：北京慧美印刷有限公司
订 阅 处：全国各地发行局
国内总发行：北京报刊发行局
邮 发 代 号：82-715
国外发行代号：M6232
刊　　号：ISSN 1006-4052 / CN11-3411/TP
广告订可证：京海工商广字 0151 号
全 年 定 价：264 元
每 期 定 价：11 元

电脑编程技巧与维护（半月刊）

每月3日出版

实用第一 智慧密集

目 次

实用第一 智慧密集

勘误：2013 年第 06 期，“软件开发与设计”栏目，“在 Spring 面向切面编程及其应用研究”一文，标题名应为“面向切面编程在 Spring 中的应用研究”。

谷歌将利用开源 WebKit 开发自主浏览器引擎 Blink

谷歌近期宣布，将在 Chromium 项目中利用开源 WebKit 引擎开发自主的 Blink 渲染引擎，并应用在 Chrome 浏览器中。

苹果最初利用 Khtml 开发了 WebKit，并应用在 Safari 浏览器中。苹果于 2005 年发布了 WebKit 的开源版本，谷歌随后在 Chrome 浏览器中应用了 WebKit。Opera 近期宣布将转向 WebKit，但该公司目前确认实际上将转向 Blink。

Blink 的推出将削弱 WebKit 在浏览器市场的影响力。到 2012 年底，WebKit 引擎在浏览器市场的份额高达 40%。

对于为何要推出 Blink，谷歌解释称，Chromium 与 WebKit 浏览器有着不同的多进程架构。谷歌工程师亚当·巴斯 (Adam Barth) 表示："过去几年中，支持多种架构给 WebKit 和 Chromium 项目增加了复杂性。"因此，这种方式对"创新的速度"产生了不利影响。

巴斯表示，谷歌做出这一决定并不容易，但谷歌认为多种渲染引擎的存在将使整个开放网络生态系统更健康。谷歌表示，在最初阶段不会进行太大调整，但将很快从代码库中删除 7 个编包系统和 7000 个文件。Blink 的策略将指导开发者何时添加新功能，同时也将删除开发商前缀。

Mozilla 和三星本周也宣布，将合作开发下一代浏览器渲染引擎 Servo。开发者对于谷歌的决定反应不一。苹果移动 Safari 团队的资深人士弗朗西斯科·托马斯基 (Francisco Tolmasky) 表示，他对这一新产品很感兴趣，而谷歌目前是 WebKit 的实际控制者。他表示："这就是软件的遭遇。目标发生改变，旧的代码和设计不再有意义，你需要重构或重新编写软件。"

新型堆叠式高性能内存 每秒读写 320GB

内存是计算机最重要的三大件之一，虽然很不起眼但却不可或缺。但长久以来内存的发展缓慢，目前桌面平台的主流还是 DDR3 标准内存。一些半导体厂商组成了名为 HMC (Hybrid Memory Cube) 的联盟，联合研发高性能内存。经过 17 个月的努力，HMC 1.0 的标准已于近期制定完成，基于该标准设计的内存性能将大幅提升，同时功耗也得到降低。

这种高性能内存采用了堆叠式技术，芯片的内部呈立体式结构，而非传统的平面式。其最大读写速度可达每秒钟 320GB，而现有的 DDR3 一般只有每秒 11GB 左右。除此之外，其功耗较传统内存也降低了 70%，更加适合移动平台。

虽然 HMC 1.0 看上去很美好，但离市场化恐怕还有一段时间。作为 HMC 成员之一的镁光计划在年内开始试制样品，而下一代的产品正在规划中，预计明年可以量产，其速度比 HMC 1.0 标准又提升了将近一倍。

微软 Windows 8.1 即将面市

近日在很多分享网站上出现了新的 Windows 8.1 版本预览版下载及大量的使用截图，看起来正式版的 Windows 8.1 已经离我们不远了。这次的 Windows 8.1 版本号为 Build 9369，虽然微软并未明确说明 Windows 8.1 将会带来哪些更新，但是通过用户试用的反馈看起来新版本将会提高系统多任务处理能力及增加多个 Metro 风格系统应用。

新版本 Windows 8.1 的后台多任务可以快速同时显示多个应用，并且将采用全新的 UI 设计元素，并且允许用户选择通过何种方式打开。在用户的后台运行两个或两个以上的应用程序时，被选择的后台应用会立即被显示到最前端。

另外，微软将文件系统的"资源管理器"和"SkyDrive 应用"合二为一。用户可以看到自己的文件系统和 SkyDrive 上所存储的内容。SkyDrive App，或者可以称它为：The Files App —— 它能够浏览用户的文件系统和 SkyDrive 云端上的内容。

同时微软似乎还改善了新系统对键盘和鼠标的支持，并且在屏幕的左侧提供了快速访问应用程序的快捷列表；另外 Windows 8.1 的搜索功能有了改进，将 Web 上的内容整合到 Windows 8 内置搜索中。也就是说，搜索一个 App 或者一个关于 Windows 8 系统的问题，可以直接得到本地的搜索结果和 Web 上的内容。

最后，Windows 8.1 新增了触摸板设置，可以让用户自行设置是否开启或关闭边界手势。

重拾开始按钮、启动至桌面

2012 年 10 月，微软推出了首个没有传统开始按钮的 Windows 系统，这一变化引发了众多质疑和批评，最严重的是，重挫了微软的整体销售。

尽管之前微软意志坚定，认为去除传统的开始按钮和开始菜单是技术进步，大众消费者总有一天会接受，但是随着时间的推移，微软似乎也开始动摇了。近日，多方消息指出，微软正在考虑听取用户的反馈，重新添加开始按钮。

在 Windows 8 的更新版本 Windows 8.1 中，微软会再度引入开始按钮，而且还会允许开机启动至桌面。爆料人对上述消息十分确定，不过也表示，不敢百分百肯定，因为到系统正式发布前一切皆有可能。

这可以看做是 Windows 8 的 B 计划，如果 Windows 8 不成功，那么微软或许就会拿出 B 计划。微软并非固执己见的家伙，他们通常勇于承认并改正错误。举例来说，在 Vista 中首度引入的用户账户管理 (UAC) 功能，就被大量用户唾弃，微软听取意见，最后在 Windows 7 中改变了 UAC 功能。

都说亡羊补牢未为晚也，如果 Windows 8.1 有传统的开始按钮和开始菜单，还允许你启动后直接来到桌面，你会不会欣然接受 Windows 8 呢？目前微软依然存在对 Windows 8.1 进行调整和改进的可能，而新版本预计将会在 6 月份的开发者大会上进行公开测试。

iBatis.Net（C#）系列三：数据库查询

张德强　祁亚玲

摘　要：查询是数据库SQL语言的核心，详细介绍了应用iBatis.Net对数据库的简单查询、条件查询、动态查询和多表查询。

关键词：iBatis.Net工具；动态查询；多表查询；数据映射

查询是数据库SQL语言的核心，SQL语言只提供唯一一个用于数据库查询的语句，即SELECT语句。用于表达SQL查询的SELECT语句是功能最强也是最复杂的SQL语句。而在实际的项目开发过程中，查询占了一个很大的比重，通常衡量一个框架的好坏也很大程度上取决于该框架对查询的灵活性和效率。下面介绍在iBatis.Net中提供的数据库查询方式。

在上篇建立的项目文件中新添加Maps/Test3.xml和Test3.aspx项，分别记录XML数据映射信息和相应的程序调用信息。

1　简单查询

获取一个表的内容，如获取DEAN.SYSUSER表的用户信息，XML数据映射配置信息为：

```
<select id="SelectSysuser"  resultMap="SysuserResult">
  SELECT *  FROM DEAN.SYSUSER
</select>
```

调用代码为：

```
protected void Button1_Click(object sender, EventArgs e)
    {
        //简单查询
        try
        {
            ISqlMapper mapper = Mapper.Instance(); //得到
//ISqlMapper 实例
            IList <iBatisTest.Domain.Sysuser > plist =
mapper.QueryForList<iBatisTest.Domain.Sysuser>("Test3
Map.SelectSysuser",null);//调用 QueryForList 方法
    if (plist ! = null && plist.Count > 0)
    {
        GridView1.DataSource = plist;
        GridView1.DataBind();
    }
    Label1.Text = "简单查询成功";
}
catch (Exception ex)
{
    Label1.Text = ex.Message;
}
```

这种查询结果返回的是整张表的所有记录，无需传入查询参数，在调用QueryForList方法时把参数置为null。考虑到系统的效率，在实际开发中，对记录数少的小表才使用。

2　条件查询

根据条件来查询结果，条件可以是一个或者多个。这种方式在实际查询中被广泛使用，是应用最多的一种查询。如根据登录用户名查询该用户信息，数据映射配置信息为：

```
<se lect  id = "SelectSysuserByUserName"
parameterClass="string" resultMap="SysuserResult">
SELECT * FROM DEAN.SYSUSER WHERE
LOGINNAME=#value#
 </select>
```

iBatis.Net通过使用#或者$符号来占位，即标识参数。在XML数据映射文件的SQL语句中引入参数有两种方式：一种是内联参数方式，即使用parameterClass，如上面的配置就采用这种方式。一种是参数映射方式，使用parameterMap。采用内联方式时，允许把属性名称、属性类型、空置方式配置在SQL语句中。

使用内联参数方式时，传入的参数有3种类型，分别为：

（1）基本参数类型，很多SQL语句在查询时只接受一个参数，如int、string，使用#value#来引用，这个value是关键字，不可变，上面的示例就采用这种方式。也可以采用#keyName#来引用，keyName为键名，注意要区分大小写。

（2）字典类型参数，使用IDictionary类型的对象作为参数。通常可以使用Hashtable。如上面配置可修改为：

```
<select id="SelectSysuserByUserName"
parameterClass="System.Collections.IDictionary" resultMap="
SysuserResult">
SELECT*FROM DEAN.SYSUSER WHERE
LOGINNAME=#LOGINNAME#
 </select>
```

（3）对象类型参数，可以是一个类或者是哈希表Hashtable，在使用哈希表时，某一个键值可以是一个列表List类型。

调用代码如下：

```
protected void Button2_Click(object sender, EventArgs e)
    {
      //条件查询
       try
       {
            string UserName = TextBox1.Text;//获取查询参
//数用户名
            ISqlMapper mapper = Mapper.Instance(); //得到
//ISqlMapper 实例
                  IList<iBatisTest.Domain.Sysuser> plist =
mapper.QueryForList<iBatisTest.Domain.Sysuser>("Test3Map.
SelectSysuserByUserName", UserName);//调用 QueryForList
//方法
          if (plist ! = null && plist.Count > 0)
          {
             GridView1.DataSource = plist;
             GridView1.DataBind();
          }
          Label1.Text = "条件查询成功";
       }
       catch (Exception ex)
       {
          Label1.Text = ex.Message;
       }
    }
```

如果采用字典类型参数传入，只需要把调用中的参数修改为：

```
Hashtable hash = new Hashtable();//声明哈希表
hash.Add("LOGINNAME ", TextBox1.Text); //获取查询
//参数用户名
```

3 动态查询

iBatis.Net 提供查询的灵活性主要体现在支持动态查询上，即可以动态地生成 SQL 语句。也只有掌握好动态查询，才能充分地感受 iBatis 框架所带来的便捷和高效。这也是很多软件公司和开发人员选择该框架的重要原因。

在开发中经常遇到这种查询，当用户没有输入查询条件时查询所有记录，如果用户输入了查询条件将根据查询条件进行查询。比如上面提到的条件查询，如果没有输入用户名信息将返回所有用户信息。这个时候就需要用到动态查询，根据参数值是否为空，生成两条不同的 SQL 语句。XML 数据映射配置信息为：

```
<select  id = "SelectSysuserDynamic1"  parameterClass = "
System.Collections.IDictionary" resultMap="SysuserResult">
  <! [CDATA[ SELECT *  FROM DEAN.SYSUSER  ]]>
  <dynamic prepend="WHERE">
   <isNotEmpty prepend="AND" property="LOGINNAME">
    <! [CDATA[ LOGINNAME = #LOGINNAME# ]]>
   </isNotEmpty>
  </dynamic>
 </select>
```

dynamic 元素用来区分 SQL 语句的动态部分，dynamic 是一个可选项，它中间可以包含任意数据的条件元素。上面的配置信息会根据输入参数 LOGINNAME 的值是否为空生成两条 SQL 语句。如果为空 SELECT * FROM DEAN.SYSUSER，如果不为空 SELECT * FROM DEAN.SYSUSER WHERE LOGINNAME = #LOGINNAME#。

调用程序代码为：

```
protected void Button3_Click(object sender, EventArgs e)
    {
       //动态查询 1
       try
       {
          Hashtable hash = new Hashtable();//声明哈希表
           hash.Add("LOGINNAME", TextBox2.Text); //获取
//查询参数用户名
           ISqlMapper mapper = Mapper.Instance(); //得到
//ISqlMapper 实例
                 IList<iBatisTest.Domain.Sysuser> plist =
mapper.QueryForList<iBatisTest.Domain.Sysuser>("Test3Map.
SelectSysuserDynamic1", hash);//调用 QueryForList 方法
          if (plist ! = null && plist.Count > 0)
          {
             GridView1.DataSource = plist;
             GridView1.DataBind();
          }
          Label1.Text = "动态查询 1 成功";
       }
       catch (Exception ex)
       {
          Label1.Text = ex.Message;
       }
    }
 }
```

在 iBatis.Net 中，动态查询的条件元素包含以下几种：二元条件元素、一元条件元素和其他条件元素。

3.1 二元条件元素

将一个属性值和静态值或另一个属性值比较，如果条件为真，元素将被包容在查询 SQL 语句中。

二元条件元素的属性：

Perpend：可被覆盖的 SQL 语句组成部分，添加在语句的前面，该属性为可选。

Property：是比较的属性，该属性为必选。

compareProperty：另一个用于和前者比较的属性（必选或选择 compareValue 属性）。

compareValue：用于比较的值（必选或选择compareProperty属性）。

二元条件元素如表1所示。

表1 二元条件元素

<isEqual>	比较属性值和静态值或另一个属性值是否相等，如果相等则查询条件有效。如： <isEqual prepend = " AND" property = " status" compareValue=" Y" > MARRIED = 'TRUE' </isEqual>
<isNotEqual>	比较属性值和静态值或另一个属性值是否不相等，如果不相等则查询条件有效。
<isGreaterThan>	比较属性值是否大于静态值或另一个属性值，如果大于则查询条件有效。如： <isGreaterThan prepend = " AND" property = " age" compareValue=" 18" > ADOLESCENT = 'FALSE' </isGreaterThan>
<isGreaterEqual>	比较属性值是否大于等于静态值或另一个属性值，如果相等等于则查询条件有效。
<isLessThan>	比较属性值是否小于静态值或另一个属性值，如果小于则查询条件有效。
<isLessEqual>	比较属性值是否小于等于静态值或另一个属性值。如： <isLessEqual prepend = " AND" property = " age" compareValue=" 18" > ADOLESCENT = 'TRUE' </isLessEqual>

二元条件元素多用在数字的区间选择上，如年龄、价格、面积等选择上面，也可以用在日期、字符串等类型的比较。如只显示ID<=10的指定ID的用户信息，如果输入值大于10则显示全部用户信息。XML数据映射配置信息为：

```
<select id = "SelectSysuserDynamic2" parameterClass = "
System.Collections.IDictionary" resultMap="SysuserResult">
  <! [CDATA[ SELECT * FROM DEAN.SYSUSER ]]>
  <dynamic prepend="WHERE">
    <isLessEqual prepend = "AND" property = "USERID"
compareValue="10">
      USERID = #USERID#
    </isLessEqual>
  </dynamic>
 </select>
```

调用代码为：

```
protected void Button4_Click(object sender, EventArgs e)
    {
        //动态查询2:二元条件元素查询
        try
        {
            Hashtable hash = new Hashtable();//声明哈希表
            int ID = 0;
            if (! string.IsNullOrWhiteSpace(TextBox3.Text))
            {
                ID = Convert.ToInt32(TextBox3.Text);
            }
            hash.Add("USERID", ID); //获取查询参数
             ISqlMapper mapper = Mapper.Instance(); //得到
//ISqlMapper 实例
                IList<iBatisTest.Domain.Sysuser> plist =
mapper.QueryForList<iBatisTest.Domain.Sysuser>("Test3Map.
SelectSysuserDynamic2", hash);//调用 QueryForList 方法
            if (plist ! = null && plist.Count > 0)
            {
                GridView1.DataSource = plist;
                GridView1.DataBind();
            }
            else
            {
                GridView1.DataSource = null;
                GridView1.DataBind();
            }
            Label1.Text = "动态查询2成功";
        }
        catch (Exception ex)
        {
            Label1.Text = ex.Message;
        }
    }
```

3.2 一元条件元素

一元条件元素检查属性的状态是否符合特定的条件。即检查属性值是否满足条件，如果满足则查询条件有效。

一元条件元素的属性和二元条件元素一样，具有prepend和property属性，其中property为必选属性。

一元条件元素如表2所示。

表2 一元条件元素

<isPropertyAvailable>	检查是否存在该属性。
<isNotPropertyAvailable>	检查是否不存在该属性。
<isNull>	检查属性是否为null。
<isNotNull>	检查属性是否不为null。
<isEmpty>	检查属性是否为空，属性的数据类型为Collection、String时检查是否为null或空，即是否为""或size() < 1。如： <isNotEmpty prepend=" AND" property=" firstName" > FIRST_NAME=#firstName# </isNotEmpty>
<isNotEmpty>	检查属性是否不为空，检查方式同上。

比如下面的配置例子：

```
<select id = "SelectSysuserDynamic3" resultMap = "
SysuserResult" parameterClass="System.Collections.IDiction
ary">
```

```
<! [CDATA[ SELECT *  FROM DEAN.SYSUSER  ]]>
<dynamic prepend="WHERE">
  <isPropertyAvailable property="SEX">
    <isNotNull property="SEX" prepend="AND">
      SEX=#SEX#
    </isNotNull>
  </isPropertyAvailable>
  <isPropertyAvailable property="STATUS">
    <isNotNull property="STATUS" prepend="AND">
      STATUS=#STATUS#
    </isNotNull>
  </isPropertyAvailable>
 </dynamic>
</select>
```

先判断传入参数集是否有 SEX 参数，如果没有则不执行 SEX=#SEX# 查询条件，再判断该参数是否为 null，不为 null 才执行查询条件。isPropertyAvailable 元素最大的好处是，如果输入的参数集不包括设置的参数时程序不会报错，直接跳过该元素设置内容。

3.3 其他元素条件

其他元素条件有两个元素，一个为 ParameterPresent，该元素检查参数对象是否存在，一个为 Iterate，该元素为遍历整个集合。

（1）ParameterPresent：ParameterPresent 元素属性只有 prepend 一个属性，表示可被覆盖的 SQL 语句组成部分，添加在语句的前面，为可选属性，如表 3 所示。

表 3 ParameterPresent 元素属性

<isParameterPresent>	检查是否存在参数对象，即如果参数类不为 NULL 则查询条件有效。如： <isParameterPresent prepend=" AND" > EMPLOYEE_TYPE = #empType# </isParameterPresent>
<isNotParameterPresent>	检查是否不存在参数对象，如： <isNotParameterPresent prepend=" AND" > EMPLOYEE_TYPE = 'DEFAULT' </isNotParameterPresent>

（2）Iterate：遍历整个集合元素，为 List 集合中的元素重复元素体的内容。

Iterate 的属性（如表 4 所示）：

表 4 Iterate 元素属性

<iterate>	遍历类型为 List 的元素。如： <iterate prepend=" AND" property=" UserNameList" open=" (" close=")" conjunction=" OR" > username=#UserNameList [] # </iterate> 注意：使用<iterate>时，在 List 元素名后面包括方括号 [] 非常重要，方括号 [] 将对象标记为 List，以防解析器简单地将 List 输出成 String。

Prepend：可被覆盖的 SQL 语句组成部分，添加在语句的前面，该属性为可选。

Property：类型为 List 的用于遍历的元素属性，该属性为必选。

Open：整个遍历内容体开始的字符串，用于定义括号，该属性为可选。

close：整个遍历内容体结束的字符串，用于定义括号，该属性为可选。

Conjunction：每次遍历内容之间的字符串，用于定义 AND 或 OR，该属性为可选。

Iterate 元素在生成 sql 语句时，标签中的内容是循环生成的，如上面的例子将会生成语句：（username =xxx1 or username=xxx2 or username=xxx 3)。该元素也经常用来动态生成 In 查询条件，如 id in （xx1,xx2,xx3,.....)，括号中的 (包括括号) 都由该元素标签生成。

比如下面的配置例子：

```
<select id="SelectSysuserDynamic4" resultMap="
SysuserResult" parameterClass="System.Collections.IDiction
ary">
  <! [CDATA[ SELECT *  FROM DEAN.SYSUSER  ]]>
  <dynamic prepend=" WHERE">
   <isPropertyAvailable property="SEX">
    <isNotNull property="SEX" prepend="AND">
     SEX=#SEX#
    </isNotNull>
   </isPropertyAvailable>
   <isNotNull prepend="And" property="USERIDLIST">
    USERID in
     <iterate  property = "USERIDLIST"  open = " ("  close = ")"
conjunction=",">
      #USERIDLIST[]#
     </iterate>
   </isNotNull>
  </dynamic>
  </select>
```

调用代码为：

```
protected void Button6_Click(object sender, EventArgs e)
    {
        //动态查询 2:其他元素条件,Iterate
try
{
    Hashtable hash = new Hashtable();//声明哈希表
    string sex = "男";
    hash.Add("SEX", sex);
    List<int> IDS = new List<int>();//声明 List 对象
    IDS.Add(1);
    IDS.Add(2);
    IDS.Add(3);
```

```
            hash.Add("USERIDLIST", IDS);
              ISqlMapper mapper = Mapper.Instance(); //得到
//ISqlMapper 实例
                    IList<iBatisTest.Domain.Sysuser> plist =
mapper.QueryForList<iBatisTest.Domain.Sysuser>("Test3Map.
SelectSysuserDynamic4", hash);//调用 QueryForList 方法
            if (plist ! = null && plist.Count > 0)
            {
               GridView1.DataSource = plist;
               GridView1.DataBind();
            }
            else
            {
               GridView1.DataSource = null;
               GridView1.DataBind();
            }
            Label1.Text = "动态查询 4 成功";
         }
         catch (Exception ex)
         {
            Label1.Text = ex.Message;
         }
        }
```

系统会根据配置信息动态地生成 In 查询条件，最终动态生成的 SQL 语句为：SELECT * FROM DEAN.SYSUSER AND SEX='男' And USERID in (1,2,3)。

4 多表查询

前面讲到的示例都是从一个表中查询记录，获取的结果也是单个对象。事实上在程序开发中，经常需要对多个表进行组合查询，返回的结果也是复杂对象。如查询用户权限信息，就需要关联用户表和权限表。

向数据库添加系统权限表 SysUserRight，脚本如下：

```
CREATE TABLE DEAN.SYSUSERRIGHT
(
   ID      NUMBER(10,0) NOT NULL ENABLE,
   USERID  NUMBER(10,0) NOT NULL ENABLE,
    RIGHTID NUMBER(10,0) NOT NULL ENABLE,
    constraint PK_SYSUSERRIGHT primary key (ID)
);
comment on column DEAN.SYSUSERRIGHT.ID is 'ID';
comment on column DEAN.SYSUSERRIGHT.USERID is '用
户 ID';
comment on column DEAN.SYSUSERRIGHT.RIGHTID is '权
限 ID';
```

有两种方式来处理这种多表查询，一种是参照单表查询，根据返回结果定制一个新类，或者直接设置返回参数为 Hashtable 表。如：

```
 <select id="MultiTable1"  resultClass="Hashtable" >
    SELECT A.*,B.RIGHTID FROM DEAN.SYSUSER A,
DEAN.SYSUSERRIGHT B WHERE A.USERID=B.USERID
 </select>
```

通过用户 ID（USERID）关联用户表（SYSUSER）和系统权限表（SYSUSERRIGHT）查询出用户信息及对应的权限信息，直接返回一个 Hashtable 表，记录了相应的信息。

第二种方式是利用 iBatis 的复杂属性来实现，在 Sysuser 类新增一个属性：

```
/// <summary>
/// 多表查询新增权限属性
/// </summary>
private int _rightid;
public int Rightid
{
   get { return _rightid; }
   set {_rightid = value;}
}
```

修改配置文件信息，在 resultMaps 部分增加一个结果映射信息，唯一号为 UserRightResult，它继承于 Test3Map.SysuserResult 结果映射，增加的配置信息如下：

```
<resultMap id="UserRightResult" class="Sysuser" extends="
Test3Map.SysuserResult">
     <result property="Rightid"  column="USERID=USERID"
select="Test3Map.SelectSysuserRight" />
</resultMap>
```

在 statements 节加入如下信息：

```
<stateme nt  id = "SelectSysuserRight"  parameterClass = "int"
resultClass="int">
   SELECT RIGHTID FROM DEAN.SYSUSERRIGHT WHERE
USERID= #USERID#
</statement>
<select id="MultiTable2" parameterClass="int"  resultMap="
UserRightResult">
   SELECT * FROM DEAN.SYSUSER  ORDER BY USERID
</select>
```

通过 XML 配置文件中 resultMap 的 result 使用“select”进行一种迭代查询，也就是将<result property=" Rightid" column="USERID=USERID" select=" Test3Map.SelectSysuserRight" />中 column 指定的一项或多项作为参数(USERID=USERID)，传入并执行指定的 select 语句 SelectSysuserRight，并将查询结果赋给 property=" Rightid "，从而实现多表查询。

该例子中实现的是 1:1 的关系查询，如果是 1:n 的关系查询，只需要 Sysuser 类增加的属性修改为 IList 类型。Statements 节点“SelectSysuserRight”的返回类修改为 resultClass="SysuserRightResult"。

通过 iBatis 复杂属性，可以非常方便地实现多表查询，但

（下转第 13 页）

基于 Delphi 的自定义编译器

于复兴 索依娜 张 昕

摘 要：应用程序内嵌自定义编译器有很强的应用价值。从一个简单的单词统计程序出发，介绍了在 Delphi 中利用词法分析器和语法分析器来创建编译器的过程，并给出了一个简单的自定义可视化编译器的制作方法。

关键词：lex 工具；yacc 分析器；词法分析；语法分析；编译器

1 问题引出

很多 Delphi 相关书籍中提到 FindDialog 与 ReplaceDialog 的用法时都举一个相同的例子：加入一个 TRichEdit 对象，在 TFindDialog 对象的 OnFind 事件中利用 TRichEdit 中所提供的 FindText 方法进行查找。那么如果在 This is a pen 这句话中查找单词 is 结果会是两处：This 中的 is 与 is 本身，若统计 is 的个数则应为 2，很显然 This 中的 is 不是我们期待的。

2 解决方案

如果想让计算机区分出 This 中的 is 不是单词 is，靠 FindText 方法的这种字符匹配是不行的，下面从编译器的词法分析角度来解决这个问题。lex（词法分析器）和 yacc（语法分析器）是特意为编写编译程序设计的，是标准的 UNIX 实用程序，

2.1 lex 和 yacc 简介

lex 是一种生成扫描器的工具。扫描器是一种识别文本中的词汇模式（或者常规表达式）的程序。一种匹配的常规表达式可能会包含相关的动作，这一动作可能还包括返回一个标记，当 lex 接收到文件或文本形式的输入时，它试图将文本与常规表达式进行匹配。它一次读入一个输入字符，直到找到一个匹配的模式。如果能够找到一个匹配的模式，lex 就执行相关的动作（可能包括返回一个标记）。另一方面，如果没有可以匹配的常规表达式，将会停止进一步的处理，并且 lex 将显示一个错误消息。按照惯例，lex 文件有.l 后缀。

Yacc 是将任何一种编程语言的所有语法翻译成针对此种语言的 yacc 语法解析器，它用巴科斯范式（BNF,Backus Naur Form）来书写。按照惯例，yacc 文件有.y 后缀。

Lex 和 Yacc 的大多数版本都是生成 C 语言程序的，接下来介绍在 Delphi 下如何使用。

安装：到 http://www.musikwissenschaft.uni-mainz.de/~ag/tply/tply.html 可以下载 tply4.1a.zip，这个软件包包含 turbo pascal 版的 Lex 和 Yacc 源程序，文档和例子。tply4.1a 可在 Linux、DOS、Win16、Win32 4 种操作系统平台以及 Free Pascal、Turbo Pascal、Borland Pascal、Delphi 等多种语言环境中编译使用。

（1）下载软件包后解压缩，可以看到在目录中有 6 个批处理文件，其中 maked32.bat 适用于 delphi。必须确保 delphi\bin 目录在系统搜索路径中，以让批处理文件找到 delphi pascal 编译程序 dcc32.exe。执行 maked32.bat，生成 lex.exe、yacc.exe。

（2）新建目录，把 lex.exe、yacc.exe、yylex.cod、yyparse.cod、lexlib.pas、yacclib.pas 拷贝至新目录中，这就构造了 Pascal Lex 和 Yacc 的最小运行环境。在原目录中，README 文件中有详细的安装信息，tply.doc 中有详细的使用说明。

2.2 应用最简单的 lex 和 yacc

下面在 Delphi 中结合 lex 和 yacc 做一个单词统计程序，如输入一个单词，统计其在此篇文章中出现的次数。

（1）词法分析：单词统计只需要扫描一遍待分析的文本，扫描到与要查找的关键字相同的匹配模式时，计数器加 1，并对任意一个单词返回一个相同标记。

（2）语法分析：接收词法分析返回的标记，保证扫描向前推进。

以下是词法分析 Tongji.l 的源代码：

```
id  [\t ]+          /* 忽略空白 */
lx  [a-zA-Z]+     /* 描述单词 */
%start
%%
{id}        ;
{lx}        begin
              chazhao();
              return(_cifayuansu);
            end;
.|\n        ;    /* 通常的默认状态 */
```

方括号“[ ]”是一个字符集合，匹配括号内的任意字符。

在示例中，接受“\t”（一个制表符）或“ ”（一个空格）。“+”意味着模式匹配加号前面的一个或多个连续的子模式的拷贝。因此这个模式描述了空白（制表符和空格的任意组合）。

模式“[a-zA-Z] +”是一种通用的模式：它表示至少包含一个字符的任意字母字符串。“-”字符在方括号之间使用时有一个特殊的含义：它指示从“-”的左边开始到“-”的右边结束的字符范围。当扫描到这个模式之一时，执行自定义chazhao () 函数 (功能：若扫描到的单词与被查找的单词相同，计数加 1)，并返回一个标记给语法分析器。

“.|\n”是默认情况语句。特殊字符“.”匹配换行符以外的任意单个字符，“\n”匹配一个换行字符。这样可以有清楚的规则来匹配所有可能的输入，不至于对出乎意料地输入字符无法分析。

以下是语法分析段 Tongji.y 的源代码：

```
%{
unit Tongji_y;
interface
uses
    SysUtils,dlib,yacclib,lexlib,db,Variants;
var  ys_count:integer; // 统计查找的数量
type
    TLexer = class(TLexerParserBase)
public
    procedure chazhao() ;
    function parse() : integer; override;
end;
    TParser = class(TLexerParserBase)
public
    lexer : TLexer;
      function parse() : integer; override;
    end;
    implementation
      uses  Unit1;
    %}
    %token _cifayuansu   //设置优先级
    %%  //语法树开始
    program_list:program
              |program_list program;
    program: _cifayuansu;
    %%  //语法树结束
    procedure TLexer.chazhao() ;
        begin
    if yytext=form1.edit1.text then  ys_count:=ys_count+1;
        end;
    {$ITongji_l.pas}
    end.
```

实际上本段中的函数可以定义在词法分析中，而本段语法分析可以没有。因为语法树在这里什么也没做，之所以这样写是为了给出一个编译器的通用格式。语法树描述的意义是：program_list:由 program 或 program_list program 组成（其本身再加上第一项），而 program 指词法分析中返回标记“_cifayuansu”的单词。

以上两段代码分别经过 Dlex 与 Dyacc 编译后，生成 Tongji_l.pas，Tongji_y.pas 文件并可直接在 Delphi 中调用。新建一个工程，加入一个 TEdit，一个 TRichEdit 和一个 TButton 组件，在 TButton 对象中添加以下代码。这样就可以单击 Button 以实现在 RichEdit 中查找 Edit 的内容并统计其数量。

```
 var
     lexer:TLexer;
     parser : TParser;
     temp_source:Tstrings;
     line:String;
   begin
 temp_source:=TStringList.Create;
 line:=self.RichEdit1.text;
     temp_source.Append(line);
   /* 为什么 C 语言区分大小写，而 Delphi 不区分，关键在这里，本来 A,a 就是两个字母你可以在这里加上一句把 line 的内容都变成大写，或小写即不区分了 */
   end;
     yyinput:=temp_source;
     yylineno := 0;
     lexer := TLexer.Create();
     parser := TParser.Create();
     parser.lexer := lexer;
     parser.parse();
    lexer.Free;
    parser.Free;
    temp_source.Free;
    showmessage(inttostr(yuansu_count));
    ys_count:=0;//计数器清零
  end;
```

程序运行结果：打开测试文件（.Txt），文本文档显示在程序中，输入要查找的单词“is”，点击查找显示查找结果为 3 次（注意 isssaw 中的 is 没有统计在内），运行结果如图 1 所示。

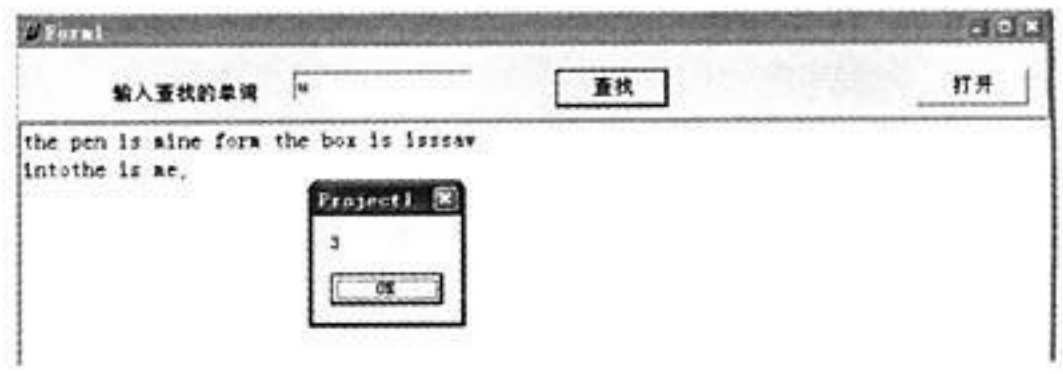


图 1　程序运行截图

3　问题升级

以上内容实际上只是编译器的雏形，虽然简单但“麻雀虽

小五脏俱全”。只须把词法、语法扩充一下即可升级为一个自定义的编译器。现假设这样一个编译器：支持 n 个 begin……end 段（类似于 Delphi）；段内可以进行变量声明（变量类型为整数如：1 a 代表声明“1”型变量 a）；变量赋值（a 5 代表变量 a 的值被赋为“5”）、加减乘除运算（包括负数、括号）。

该编译器的词法为：

```
num ([0-9]+|([0-9]*\.[0-9]+)([eE][-+]?[0-9]+)?)
//该模式匹配实型数据
lx  [0-9]+              //匹配整形数据
id   [A-Za-z]([A-Za-z]|[0-9])* //匹配变量名
%start
%%
var result : integer;
{lx}  return(_LX)
{num}   return(NUM)
{id}    return(ID);
"begin"   return(_BEGIN);
"end"   return(_END);
+   returnc('+');
-   returnc('-');
*   returnc('*');
/   returnc('/');
"("    returnc('(');
")"    returnc(')');
.|\n   ;
```

词法扫描时各模式及符号返回相应的标识。

该编译器的语法树为：

```
%token _BEGIN _END
%token <Integer> _LX
%token <Real> NUM
%token <string> ID
%token '(' ')'
%type <Real> expr
%left '+' '-' _OR
%left '*' '/' _AND
%right UMINUS
%%
program_list:program
        |program_list program;
program: _BEGIN{yyoutput.append('开始');}
         statement_list
        _END{yyoutput.append('结束');
                deleteFHB(); };
// 程序以 begin 开始,end 结束
//中间是语句列表"statement_lis"
statement_list:statement
          |statement_list statement;
statement: jbsm
        |ID expr{ searchFHB($1);
          yyfhb['vvalue']:=floattostr($2);
          yyfhb.next;
          yyoutput.append(floattostr($2));};
// statement 可以是声明"qjsm"
//或赋语句"ID expr"
jbsm: _LX ID {insertFHB($2);}
        | qjsm _LX ID { insertFHB($3);};
// jbsm 声明一个变量,或声明一个变//量列表
expr : NUM
       |expr '+' expr {$$ := $1 + $3;}
       |expr '-' expr {$$ := $1 - $3;}
       |expr '*' expr {$$ := $1 * $3;}
       |expr '/' expr {$$ := $1 / $3;}
       |'(' expr ')'  {$$:=$2;     }
       |  '-' expr { $$ := -$2;    }
         %prec UMINUS
  |ID{ searchFHB($1);
$$:=strtofloat(yyfhb['vvalue']) ;};
//expr 中可以是一个实数,表达式,或变量。
    %%
```

%token 声明标记并可指定类型；%type 声明非终结符号 expr 并设置其类型<Real>；%left，%right 设置运算符的结合方向（左，右）并且指定优先级越向下越高。UMINUS 代表一元减号的伪标记，右结合且优先级最高，即取负数。每次词法分析程序读取输入中的变量时，要在符号表中查找这个变量，以取得它的值，找到则修改变量，找不到则报错 (变量要先声明在使用)，当然声明变量时要把变量存入符号表。在程序中有多个 begin……end 段，所以当退出 end 时要把符号表中的变量删除。(begin……end 段中的变量为局部变量，在其他段里无效)。在这个例子中，通过 Delphi 中的 TTable 组件来实现符号表，用 TTable 的 savetofile 方法将符号表保存成文件，然后用 TTable 的 locate 方法进行查询，Insert 方法进行插入，DeleteRecords 方法进行删除。把这些方法封装在自定义的 searchFHB ()，insertFHB ()，deleteFHB () 函数中。函数的实现展开如下：

```
    procedure  TParser.insertFHB (vids:String;vtype:string;
vvalue:String;vlevel:integer) ;
  begin
      yyfhb.Insert;
      yyfhb['vids']:=vids;
      yyfhb['vtype']:=vtype;
      yyfhb['vvalue']:=vvalue;
      yyfhb['vlevel']:=vlevel;
      yyfhb.Next;

  end;// 变量插入符号表

   function  TParser.searchFHB  (vids:String;level:integer) :
boolean;
begin searchFHB:=yyfhb.Locate('vids;vlevel',VarArrayOf([vids,
level]),[locaseInsensitive]);
```

```
end;
// 在符号表中查找变量
procedure TParser.deleteFHB(level:integer) ;
begin
      yyfhb.filter:= 'vlevel='+inttostr(level);
      yyfhb.filtered := true;
      while (not yyfhb.eof) do
      yyfhb.DeleteRecords(arcurrent);
      yyfhb.Filtered:=false;
 end;
// 符号表中变量删除(局部有效)
```

4 结语

利用写自动机来实现编译器，非常麻烦。文中利用 lex 与 yacc 工具在 Delphi 中能很容易实现自定义编译器。特点是利用了 Delphi 中的可视化控件，一改已往其他书籍中用 lex 与 yacc 实现的 Dos 界面效果。可以把自定义编译器内嵌在应用程序的各个地方，有很强的实用价值。

参考文献

[1] Jobn R.Levine,Tony Mason. lex 与 yacc. 2 版. 机械工业出版社，2003.

（收稿日期：2012-11-12）

（上接第 9 页）

这个方法给我们带来便利的同时，也带来了两个问题。首先，创建包含大量对象的列表可能会消耗大量的内存。其次，这种方法会导致数据库的 I/O 问题，其原因就是所谓的“N+1”查询现象。对于主从表（也称为父子表）的查询，特别容易产生 N+1 查询问题，N+1 查询问题是由于试图加载多个父记录（比如 User）的子记录（Right）而引起的。在查询父记录时，只需要 1 条语句，假设返回 N 条记录，那么就需要再执行 N 条语句来查询子记录，引发所谓的“N+1 查询”。

解决 N+1 查询问题可以通过延迟加载（lazy loading）来实现，它将加载过程打散为一些更小的过程。在父子表查询过程中，对子表的查询往往不需要和父表一起加载，例如，系统的用户管理，打开一个用户信息页面时显示的是用户信息列表，当点击一个用户时，才需要加载该用户的权限信息。这种情况就特别适合使用延迟加载，每次都仅查询一个列表。使用延迟加载的时候还需要特别注意，使用的动态代理的对象的所有方法和属性都必须是 virtual 类型。

要实现延迟加载，只需要在配置文件里面加入 lazyLoad="true" 属性就可以了。通过延迟加载，它能提高查询的效率，但并没有真正解决数据库 I/O 问题，在最坏的情况下，它对数据库的访问次数与非延迟加载的时候是一样的。如何真正解决 N+1 查询问题呢？iBatis 提供了连接语句（join）方式来完全避免 N+1 查询的出现。

修改配置文件信息，在 resultMaps 部分增加如下配置：

```
<resultMap id="SysuserRightResult" class="SysuserRight">
   <result property="ID" column="ID"/>
   <result property="Userid" column="USERID" />
   <result property="Rightid" column="RIGHTID"/>
</resultMap>
<resultMap  id = "SysuserJoinResult"  class = "SysuserJoin"
extends="Test3Map.SysuserResult" groupBy="Userid">
   <result  property = "RightList"  resultMapping = "Test3Map.
SysuserRightResult" />
</resultMap>
```

这个配置使用了 relultMap 的 resultMapping 属性，该属性用在如果一个数据类的属性本身不是基元数据类型，而是一个复杂数据类型的场景。这个时候就不能用一个简单的 result 元素来表示，必须给它一个完整的 resultMap。resultMapping 的值指明 RightList 属性由结果映射集 SysuserRightResult 所表示的复杂数据类型表示。因为用户和权限信息是一对多的关系，在主表的结果映射上加入 groupBy=" Userid" 属性。

注意用户类 SysuserJoin 的权限属性 RightList 的定义，它是由权限类 SysuserRight 组成的一个列表。RightList 定义如下：

```
  // 多表查询新增权限列表属性
  private IList<SysuserRight> _rightlist;
  public IList <SysuserRight> RightList
   {
     get { return _rightlist; }
     set { _rightlist = value; }
     }
```

在 statements 节加入如下信息：

```
<select id=" MultiTable3"  resultMap=" SysuserJoinResult"  >
   select A.*,B.* FROM DEAN.SYSUSER A LEFT JOIN DEAN.
SYSUSERRIGHT B ON A.USERID=B.USERID  </select>
```

再通过程序的调用，就不会出现 N+1 查询问题。

5 结语

以上程序在 VS2012 (C#) +Oracle11gR2+Windows7 (64 位) 调试通过，附件(略)的例子程序中有详细的配置信息和程序调用代码。

（收稿日期：2013-01-09）

PHPExcel 文件读写

丁月光

摘　要：介绍了 PHPExcel 的应用背景和常见用法，给出了一个电子商务网站中用 PHP 实现库存盘点应用场景的 MySQL 数据库与 Excel 文件的数据交互实例，并对其他可应用场合做了简要介绍。

关键词：PHP 语言；MySQL 数据库；PHPExcel 类

1　PHPExcel 概述

在 PHP 开发工作中，经常会有 MySQL 数据库与其他文件进行导入导出等交互的开发需求，常见的有 txt、csv 和 Excel 等形式的文件，其中 txt、csv 只能实现特定格式的纯数据读写，而 Excel 是最常见的桌面电子表格应用，PHP 实现与 Excel 文件读写交互的功能较为困难。

为了实现与 Excel 文件的交互，Maarten Balliauw 等人成立了一个团队，设计了 PHPExcel 这样一个强大的开源 Excel 电子表格类，它功能极其强大，可以导出 xls、xlsx（Excel2007 格式）、html、pdf、csv 等多种格式的文件，还可以对 xml 模板进行编辑然后保存，更难得的是可以在 Excel 里面设置图片、表格、字体大小、颜色等非常具体的格式，对 Excel 文件的操作有点无所不能。

PHPExcel 虽然可以导出 pdf 格式的文件，但其对中文的支持不够好，不推荐使用。

PHPExcel 官方网站地址为 http://phpExcel.codeplex.com/，目前最高版本为 1.7.7。

2　常见用法

```
//包含 PHPExcel 类库
require_once 'phpExcel/PHPExcel/IOFactory.php';
//打开 Excel 文件
$objPHPExcel = PHPExcel_IOFactory::load($fileName);
//对文档的操作,设置文档基本属性
$objProps = $objExcel->getProperties();
//创建人
$objProps->setCreator("Bill");
//最后修改人
$objProps->setLastModifiedBy("Bolm");
//标题
$objProps->setTitle("Office XLS Test Document");
//题目
$objProps->setSubject("Office XLS Test Document, Demo");
//描述
$objProps ->setDescription ("Test document, generated by PHPExcel.");
//关键词
$objProps->setKeywords("office Excel PHPExcel");
//类别
$objProps->setCategory("Test");
//设置当前的 sheet 索引,用于后续的内容操作。一般只有在
//使用多个 sheet 的时候才需要显示调用。
//缺省情况下,PHPExcel 会自动创建第一个 sheet 被设置 SheetIndex=0
$objExcel->setActiveSheetIndex(0);
//获取当前活动工作表
$objActSheet = $objPHPExcel->getActiveSheet();
//设置当前活动 sheet 的名称
$objActSheet->setTitle('测试 Sheet');
//设置单元格内容
//由 PHPExcel 根据传入内容自动判断单元格内容类型
$objActSheet->setCellValue('A1', '字符串内容'); // 字符串
//内容
$objActSheet->setCellValue('A2', 26);  // 数值
$objActSheet->setCellValue('A3', true);  // 布尔值
$objActSheet->setCellValue('A4', '=SUM(A2:A2)'); // 公式
//显式指定内容类型
$objActSheet->setCellValueExplicit('A5', '847475847857487584', PHPExcel_Cell_DataType::TYPE_STRING);
//合并单元格
$objActSheet->mergeCells('B1:C22');
//分离单元格
$objActSheet->unmergeCells('B1:C22');
//设置单元格样式
//设置宽度
$objActSheet ->getColumnDimension ('B') ->setAutoSize (true);
$objActSheet->getColumnDimension('A')->setWidth(30);
//设置字体
$objFontA5 = $objStyleA5->getFont();
$objFontA5->setName('Courier New'); //字体名称
```

```
$objFontA5->setSize(10); //字号大小
$objFontA5->setBold(true); //斜体
$objFontA5->setUnderline(PHPExcel_Style_Font::UNDERLINE_SINGLE); //下划线
$objFontA5->getColor()->setARGB('FF999999'); //字体颜色
//设置对齐方式
$objAlignA5 = $objStyleA5->getAlignment();
$objAlignA5 ->setHorizontal (PHPExcel_Style_Alignment::HORIZONTAL_RIGHT); //居右对齐
$objAlignA5->setVertical(PHPExcel_Style_Alignment::VERTICAL_CENTER); //垂直居中对齐
//设置边框
$objBorderA5 = $objStyleA5->getBorders();
$objBorderA5->getTop()->setBorderStyle(PHPExcel_Style_Border::BORDER_THIN); //边框顶线类型
$objBorderA5->getTop()->getColor()->setARGB('FFFF0000'); //边框颜色
$objBorderA5->getBottom()->setBorderStyle(PHPExcel_Style_Border::BORDER_THIN);//底线类型
$objBorderA5->getLeft()->setBorderStyle(PHPExcel_Style_Border::BORDER_THIN); //左边线类型
$objBorderA5->getRight()->setBorderStyle(PHPExcel_Style_Border::BORDER_THIN); //右边线类型
//设置填充颜色
$objFillA5 = $objStyleA5->getFill();
$objFillA5->setFillType(PHPExcel_Style_Fill::FILL_SOLID);
$objFillA5->getStartColor()->setARGB('FFEEEEEE');
//从指定的单元格复制样式信息.
$objActSheet->duplicateStyle($objStyleA5, 'B1:C22');
//添加图片
$objDrawing = new PHPExcel_Worksheet_Drawing();
$objDrawing->setName('Obama');
$objDrawing->setDescription('Image inserted by Obama');
$objDrawing->setPath('./zssezhil.gif');
$objDrawing->setHeight(36);
$objDrawing->setCoordinates('C23');
$objDrawing->setOffsetX(10);
 $objDrawing->setRotation(15);
$objDrawing->getShadow()->setVisible(true);
$objDrawing->getShadow()->setDirection(36);
$objDrawing->setWorksheet($objActSheet);
//添加一个新的 worksheet
$objExcel->createSheet();
$objExcel->getSheet(1)->setTitle(' 测试 2');
//保护单元格
$objExcel->getSheet(1)->getProtection()->setSheet(true);
$objExcel->getSheet(1)->protectCells('A1:C22', 'PHPExcel');
//输出内容
$outputFileName = "output.xls";
//到文件
$objWriter->save($outputFileName);
//到浏览器
header("Content-Type: application/force-download");
header("Content-Type: application/octet-stream");
header("Content-Type: application/download");
header('Content-Disposition:inline;filename="'.$outputFileName."");
header("Content-Transfer-Encoding: binary");
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");
header ("Cache -Control: must -revalidate, post -check =0, pre-check=0");
header("Pragma: no-cache");
$objWriter->save('php://output');
```

上面的几个例句也可以用这样的写法：

```
//设置单元格的值
$objPHPExcel->getActiveSheet()->setCellValue('A1', 'String');
$objPHPExcel->getActiveSheet()->setCellValue('A2', 12);
$objPHPExcel->getActiveSheet()->setCellValue('A3', true);
$objPHPExcel ->getActiveSheet ()->setCellValue ('C5', '=SUM(C2:C4)');
$objPHPExcel->getActiveSheet()->setCellValue('B8', '=MIN(B2:C5)');
//合并单元格
$objPHPExcel->getActiveSheet()->mergeCells('A18:E22');
//分离单元格
$objPHPExcel->getActiveSheet()->unmergeCells('A28:B28');
//保护 cell
$objPHPExcel->getActiveSheet()->getProtection()->setSheet(true);
$objPHPExcel->getActiveSheet()->protectCells('A3:E13', 'PHPExcel');
//设置宽 width
$objPHPExcel->getActiveSheet()->getColumnDimension('B')->setAutoSize(true);
$objPHPExcel->getActiveSheet()->getColumnDimension('D')->setWidth(12);
//设置 font
$objPHPExcel->getActiveSheet()->getStyle('B1')->getFont()->setName('Candara');
$objPHPExcel->getActiveSheet()->getStyle('B1')->getFont()->setSize(20);
$objPHPExcel->getActiveSheet()->getStyle('B1')->getFont()->setBold(true);
$objPHPExcel->getActiveSheet()->getStyle('B1')->getFont()->setUnderline
//处理中文输出问题
```

需要将字符串转化为 UTF-8 编码，才能正常输出，否则中文字符将输出为空白，如下处理：

```
$str = iconv('gb2312', 'utf-8', $str);
```

3 电子商务网站库存盘点

假定一个电子商务网站，为工作人员方便，每天从网站数

据库导出 Excel 格式的电子表格文件作为工作人员存档或者在此基础上进行库存盘点。库存盘点后，修改 Excel 文件上传到网站更新网站数据库信息。

3.1 网站数据库设计

本例程仅为演示所用，因此数据库字段无需考虑过多，只设计一个表，名为“goods”，id、商品代码、品名、品牌、型号、单价、数量即可：

```
--
-- 表的结构 `goods`
--
CREATE TABLE IF NOT EXISTS `goods` (
  `id` int(5) NOT NULL AUTO_INCREMENT,
  `itemcode` varchar(10) NOT NULL,
  `itemname` varchar(20) NOT NULL,
  `itemband` varchar(18) NOT NULL,
  `itemmodel` varchar(20) NOT NULL,
  `price` float NOT NULL,
  `number` int(8) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=gbk COMMENT=
'商品库存表' AUTO_INCREMENT=7 ;
```

添加数行演示数据：

```
INSERT INTO `goods` (`id`, `itemcode`, `itemname`,
`itemband`, `itemmodel`, `price`, `number`) VALUES
(1, '1251DONC', '移动电源', 'YB', 'YB002', 180, 280),
(2, 'YLB312G', '智能手机', 'STCSG', 'G67', 1860, 56),
(3, 'H821LSF', 'TF 卡', '明和', '8G', 25, 2200),
(4, 'JK1287YS', '移动硬盘', 'WXS', '500G', 500, 26),
(5, 'NG264LA', '数码相机', '名宝', 'YOV', 1800, 120),
(6, 'LKNV183', '理线带', 'LKII', 'CLS', 6, 2120);
```

3.2 Excel 文件设计

对应数据库的设计，Excel 文件也仅需一个工作表，第一行填充 5 列，分别为序号、商品代码、品名、品牌、规格型号、单价、数量。第二行开始空白即可。

3.3 例程文件 Excel.php

Excel 文件上传数据库是一个普通的文件上传应用，关系不大，这里就不做详细演示了。假定文件已经上传成功，存储于 Excel.php 文件同目录下，名为“库存.xls”。

```
<?php
//数据库类定义
class db_mySQL {
    var $querynum = 0;
    var $link;
    var $histories;
    var $time;
    var $tablepre;
    function connect($dbhost, $dbuser, $dbpw, $dbname =
'', $dbcharset, $pconnect = 0,
$tablepre='', $time = 0) {
        $this->time = $time;
        $this->tablepre = $tablepre;
        if($pconnect) {
            if (! $this->link = mysql_pconnect($dbhost,
$dbuser, $dbpw)) {
    $this->halt('Can not connect to MySQL server');
            }
        } else {
            if (! $this->link = mysql_connect($dbhost,
$dbuser, $dbpw, 1)) {
        $this->halt('Can not connect to MySQL server');
            }
        }
        if($this->version() > '4.1') {
            if($dbcharset) {
        mysql_query ("SET character_set_connection = ".
$dbcharset."
character_set_results = ". $dbcharset.", character_set_client =
binary", $this->link);
            }
            if($this->version() > '5.0.1') {
        mysql_query("SET sql_mode=''", $this->link);
            }
        }
        if($dbname) {
            mysql_select_db($dbname, $this->link);
        }
    }
        function fetch_array ($query, $result_type =
MYSQL_ASSOC) {
        return mysql_fetch_array($query, $result_type);
    }
    function result_first($sql, &$data) {
        $query = $this->query($sql);
        $data = $this->result($query, 0);
    }
    function fetch_first($sql, &$arr) {
        $query = $this->query($sql);
        $arr = $this->fetch_array($query);
    }
    function fetch_all($sql, &$arr) {
        $query = $this->query($sql);
        while($data = $this->fetch_array($query)) {
            $arr[] = $data;
        }
    }
    function cache_gc() {
        $this->query ("DELETE FROM {$this->tablepre}
sqlcaches WHERE expiry<$this-
>time");
    }
```

```
    function query($sql, $type = '', $cachetime = FALSE) {
                        $func = $type == 'UNBUFFERED'
&& @function_exists('mysql_unbuffered_query') ?
'mysql_unbuffered_query' : 'mysql_query';
if(!($query = $func($sql, $this->link)) && $type != 'SILENT')
{
            $this->halt('MySQL Query Error', $sql);
        }
        $this->querynum++;
        $this->histories[] = $sql;
        return $query;
    }
    function affected_rows() {
        return mysql_affected_rows($this->link);
    }
    function error() {
return (($this->link) ? mysql_error($this->link) : mysql_error());
    }
    function errno() {
        return intval (($this->link) ? mysql_errno ($this->
link) : mysql_errno());
    }
    function result($query, $row) {
        $query = @mysql_result($query, $row);
        return $query;
    }
    function num_rows($query) {
        $query = mysql_num_rows($query);
        return $query;
    }
    function num_fields($query) {
        return mysql_num_fields($query);
    }
    function free_result($query) {
        return mysql_free_result($query);
    }
    function insert_id() {
        return ($id = mysql_insert_id($this->link)) >= 0 ?
$id : $this->result
($this->query("SELECT last_insert_id()"), 0);
    }
    function fetch_row($query) {
        $query = mysql_fetch_row($query);
        return $query;
}
function fetch_fields($query) {
        return mysql_fetch_field($query);
}
function version() {
        return mysql_get_server_info($this->link);
}
function close() {
        return mysql_close($this->link);
    }
    function halt($message = '', $sql = '') {
        echo 'run_sql_error' . $message.'<br /><br />'.
$sql.'<br /> '.mysql_error();
    }
}
//数据库连接参数定义
//MySQL 数据库地址
$dbhost = 'localhost';
//数据库用户名
$dbuser = 'root';
//数据库密码
$dbpwd = '';
//数据库名称
$dbname = 'shop';
//数据库编码格式
$dbcharset = 'gbk';
//初始化数据库连接
$db = new db_mySQL();
$db ->connect ($dbhost, $dbuser, $dbpwd, $dbname,
$dbcharset);
//包含 PHPExcel 类库
require_once 'phpExcel/PHPExcel/IOFactory.php';
//上传到服务器上的 Excel 文件
$fileName = ' 库存.xls';
//打开 Excel 文件
$objPHPExcel = PHPExcel_IOFactory::load($fileName);
//设定当前工作表为第一个
$cursheet=$objPHPExcel->setActiveSheetIndex(0);
//读取 excle 文件内容,并存入数据库
if($_GET['action'] == "in"){
?>
< ! DOCTYPE html PUBLIC " -//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http -equiv = "Content -Type" content = "text/html;
charset=gb18030" />
<title>Excel 与 MySQL 交互示例</title>
</head>
<body>
<?php
    //取 Excel 文件内数据行数。Excel 文件格式固定,无需使
//用$cursheet->getHighestColumn()取最大列数
    $row = $cursheet->getHighestRow();
    echo "<div align=\"center\">库存商品一览表</div>\n";
    echo " <table align =\"center\" cellpadding =\"1\"
cellspacing=\"1\" border=\"1\">\n";
    echo "<tr><td>序号</td><td>商品代码</td><td>品名<
/td><td>品牌</td><td>规格型号
```

```
</td><td>单价</td><td>数量</td></tr>\n";
    for($currow=2;$currow<=$row;$currow++){
        //循环从每行每列的单元格中取值
        //第一行为标题行,数据行从第二行开始
        //网页和数据库都采用 gbk 编码,需要将 Excel 文件
//的内容转码
        $curGoods['id'] = iconv('utf-8','gbk',$cursheet->
getCell('A'.$currow)- >getValue());
        $curGoods ['itemcode'] = iconv ('utf -8','gbk',
$cursheet->getCell('B'.$currow)- >getValue());
        $curGoods ['itemname'] = iconv ('utf -8','gbk',
$cursheet->getCell('C'.$currow)- >getValue());
        $curGoods ['itemband'] = iconv ('utf -8','gbk',
$cursheet->getCell('D'.$currow)- >getValue());
        $curGoods ['itemmodel'] = iconv ('utf -8','gbk',
$cursheet->getCell('E'. $currow)->getValue());
        $curGoods ['price'] = iconv ('utf -8','gbk',
$cursheet->getCell('F'.$currow)- >getValue());
        $curGoods ['number'] = iconv ('utf -8','gbk',
$cursheet->getCell('G'.$currow)- >getValue());
        $db ->query ("replace into goods (id,itemcode,
itemname,itemband,itemmodel,price,number) values ('" .
$curGoods ['id'] . "', '". $curGoods ['itemcode'] . "', '" .
$curGoods['itemname'] . "', '" . $curGoods['itemband'] . "','" .
$curGoods ['itemmodel'] . "','" . $curGoods ['price'] . "','" .
$curGoods['number'] . "')");
            echo "<tr><td>" . $curGoods ['id'] . "</td><td>" .
$curGoods ['itemcode'] . " </td ><td > " . $curGoods
['itemname'] . "</td><td>" . $curGoods['itemband'] . "</td><
td>" . $curGoods ['itemmodel'] . "</td><td>" . $curGoods
['price'] . "</td><td>" . $curGoods['number']. "</td></tr>\n";
    }
    echo "</table>\n";
    echo "<div align=\"center\"><br/>\n 上述数据已成功存入
数据库。</div>\n";
    echo "</body>\n";
    echo "</html>";
}
//从数据库中读取数据,写入 Excel 文件导出
if($_GET['action'] == "out"){
    $result = $db->query("select * from goods");
    $objPHPExcel->getProperties()->setCreator("电商网");
    $objPHPExcel->getProperties()->setTitle("库存清单");
//从第二行开始循环每行每列写入 Excel 文件的每个单元格
    $i = 2;
    while($curGoods = $db->fetch_array($result)){
        $cursheet->setCellValue('A'.$i, iconv('gbk','utf-8',
$curGoods['id']));
        $cursheet->setCellValue('B'.$i, iconv('gbk','utf-8',
$curGoods['itemcode']));
        $cursheet->setCellValue('C'.$i, iconv('gbk','utf-8',
$curGoods['itemname']));
        $cursheet->setCellValue('D'.$i, iconv('gbk','utf-8',
$curGoods['itemband']));
        $cursheet->setCellValue('E'.$i, iconv('gbk','utf-8',
$curGoods['itemmodel']));
        $cursheet->setCellValue('F'.$i, iconv('gbk','utf-8',
$curGoods['price']));
        $cursheet->setCellValue('G'.$i, iconv('gbk','utf-8',
$curGoods['number']));
        $i++;
    }
    //输出文件下载的 header 定义
    header('Content-Type: application/vnd.ms-Excel');
    header ('Content-Disposition: attachment;filename="库
存清单.xls"');
    header('Cache-Control: max-age=0');
        $objWriter = PHPExcel_IOFactory::createWriter
($objPHPExcel, 'Excel5');
    $objWriter->save('php://output');
}
?>
```

3.4 程序运行

使用方法一（如图 1 所示），自 Excel 文件中读取数据存入数据库，访问方式为 http://localhost/Excel.php?action=in。

使用方法二（如图 2 所示），从数据库中读取数据输出为 Excel 文件下载，访问方式为 http://localhost/Excel.php?action=out。

图 1 自 Excel 文件中读取数据存入数据库

图 2 从数据库中输出 Excel 文件下载

3.5 结语

全国专业技术人员计算机应用能力考试（职称计算机考试）提供了现场报名模式 Excel 文件批量报名的功能，由报名工作人员集中录入后导入其考务系统，但其网上报名系统应用非常不便，在另行开发网上报名系统时，考察了多种办法实现与其考务系统对接，最后选定使用 PHPExcel 类库，实现了网上报名数据库与标准格式 Excel 文件的读写，完美地实现了网上报名功能，和原有考务系统无缝结合，方便了考点的报名工作。

（收稿日期：2012-12-10）

C#WebForm 中的相互传值和操作

黎 明

摘 要：详细论述了在.Net 平台下进行 WebForm 窗体之间相互传值和操作的方法。

关键词：C# 语言；WebForm 控件；窗体传值

1 引言

程序员在.Net 平台下进行 WebForm 的应用系统的开发过程中，经常会遇到窗体之间的相互传值或者其他的调用操作，这是在.Net 平台上编写 Winform 程序的基础，网络上问及这方面的帖子特别多，下面举例并将具体的方法做了总结。

2 编程

设计如图 1 所示的两个窗体。

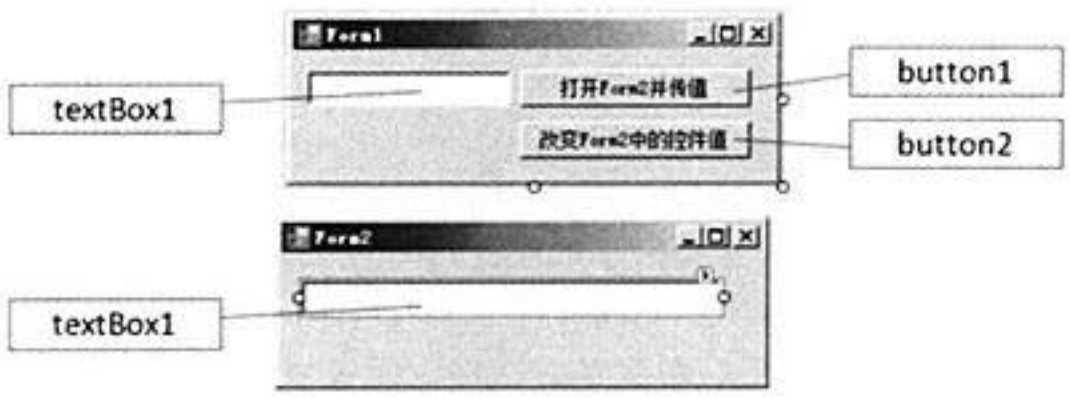


图 1 窗体设计

（1）可以通过改变窗体控件的访问属性来进行操作。

Form1 中 button1 的功能代码相对简单：

```
Form2 Frm2 = new Form2();
Frm2.Show();
Form2.textbox1.text=textbox1.text;
```

可是最后的代码有问题，系统会提示“窗体调用.Form2”并不包含“textbox1”的定义”，看 Form2 中的定义文件中，有“private System.Windows.Forms.TextBox textBox1;”说明 Form2 窗体中的 textbox1 是私有的，对外不能访问，怎么办？改动访问属性，将“Private”改为“internal”（即在本程序集里可以访问）再运行，系统正常了。

（2）可以定义一个传递对象并且在窗体的构造函数中接收传递对象来处理窗体的传值操作。

系统将窗体控件的访问属性默认为私有，这样做是遵循了面向对象的封装原则，所以，还是将“internal”改回“Private”，那上面的问题怎么解决？

既然是打开窗体就传值，那么在 Form2 中声明一个构造函数，然后在 Form2 窗体初始化的时候就传值就可以了，在 Form2 中声明新的构造函数：

```
public Form2(string SValue)
{
    this.textBox1.Text = SValue;
}
```

在 Form1 窗体的 button1 按钮的点击事件中写如下代码：

```
Form2 Frm2 = new Form2(textBox1.Text);
Frm2.Show();
```

代码没有问题，运行时系统报告构造函数错误，原来 Form2 (string SValue) 的 textbox1 对象为空，那么就是说该代码运行于初始化事件之前，所以改动代码如下：

```
public Form2(string SValue)
{
    InitializeComponent();//加入初始化事件
    this.textBox1.Text = SValue;
}
```

运行，系统正常了。

（3）根据封装原则，通过属性来封装对象内的字段操作提高安全性，检索 Application.OpenForms 可以对系统目前打开的窗体进行操作。

现在处理 Form1 中的 button2 事件，它的作用是在 Form2 窗体打开的情况下改变该窗体中的 textbox1 的值。

写代码时必须找到打开的窗体然后再赋值，因为对 Frm2 定义只在 button1_Click 方法体中存在，所以无法在本方法体中利用 Form2，那么是不是在把 Form2 的作用域延伸到整个 Form1 窗体（在 Form1 中定义）就可以利用 Form2 了。但是这个没有解决问题，关键在于 Form2 中的 textbox1 是私有的，所以，没有必要在 Form1 中定义对 Form2 的引用。

找到目前打开的 Form2 窗体的实例对象，可以应用 Application 对象，它包含了一个集合对象 OpenForms，通过检索它可以找到目前打开的窗体。

textbox1 既然是私有的，那么可以通过定义属性来操作它，如下：

```
public string Textbox1
{
    get { return textBox1.Text; }
```

```
        set { textBox1.Text = value; }
    }
```

那么 Form1 中的 button2 的点击事件代码可以这样写了：

```
((Form2)Application.OpenForms ["Form2"]).Textbox1 = textBox1.Text;
```

运行系统，正常。

问题又来了，当打开多个 Form2 实例对象窗体时，上面的代码只改变最先打开的那个，其他的没有反应。

因为 Application.OpenForms [" Form2"]) 是按 Form2 名称检索，检索到后就返回了，所以只有最先打开的 Form2 对象接收了操作，改动 Form1 中的 button2 的点击事件代码如下：

```
foreach (Form Frm in Application.OpenForms)
{
        if (Frm.Text == "Form2")
        {
            ((Form2)Frm).Textbox1 = textBox1.Text;
        }
}
```

这样就可以对所有打开的 Form2 对象实例进行相同的操作了。

(4) 可以通过委托和事件来进行窗体之间的传值操作。

窗体传值是一个事件的改变触发另外一个方法或者动作，对于此类应用一般通过委托来实现。

在 Form2 中写下如下代码：

```
public delegate void MyDelegateChange(string Str);//定义委托
public MyDelegateChange MyChange;//声明
//定义动作
private void MyChangeFunction(string Str)
{
    textBox1.Text = Str;
}
```

在 Form2 的 button2 的点击事件中写下如下代码：

```
MyChange = new MyDelegateChange(MyChangeFunction);
```

在 Form1 的 button2 的点击事件中写下如下代码：

```
Form2 Frm2 = new Form2();
foreach (Form Frm in Application.OpenForms)
{
    if (Frm.Text == "Form2")
    {
        Frm2 = (Form2)Frm;
    }
}
Frm2.MyChange(textBox1.Text);
```

运行系统，正常传值。

还可以通过委托定义事件来处理。

在 Form2 中写下人下代码：

```
public delegate void MyDelegateChange(string Str);//定义委托
public event MyDelegateChange OnMyChange;//声明事件
public void Form2_OnMyChange(string Str)
{
    //事件动作
    textBox1.Text = Str;
}
```

在 Form1 的 button2 的点击事件中写下如下代码：

```
Form2 Frm2 = new Form2();
Frm2.Form2_OnMyChange(textBox1.Text);
```

运行，系统正常传值。

(5) 可以通过检索集合对象来进行窗体之间相互灵活的操作。

其实，上面是秉承面向对象的原则来处理，看起来麻烦，如下方法可以随意地达到目的：

```
Form Frm2 = new Form2();
foreach (Form Frm in Application.OpenForms)
{
    if (Frm.Text == "Form2")
    {
        Frm2 = Frm;
    }
}
Frm2.Controls["textbox1"].Text = textBox1.Text;
```

在写代码过程中，通过上面的方式只能操作控件共有的属性和方法，如果是控件特有的，可以通过引用的方式进行处理，假设 Form2 上面有个 CheckBox1 的控件，现在在 Form1 的 Button2 的点击事件中操作它，可以按如下的方式：

```
CheckBox CB1=new CheckBox();
CB1 = Frm2.Controls["checkBox1"] as CheckBox;
CB1.Checked = true;
```

比如，现在 Form2 窗体上有个名为 button1 的按钮，希望触发它的点击事件，可以这样写：

```
((Button)Frm2.Controls["button1"]).PerformClick();
```

3 结语

网络上，对于对话框窗口的传值的问题也很多，解决办法是如果想获取一个窗口的返回值，直接调用就可以。

另外，还可以通过 API 的调用来进行窗体之间的相互操作，限于篇幅这里不再列出。

(收稿日期：2013-01-11)

MIDI 编辑器开发

江 洪

摘 要：结合 MIDI 文件的结构，使用 VC6.0 开发了 MIDI 编辑器程序。该程序可以完成基本的 MIDI 文件的生成和读取功能。通过本程序可以了解 MIDI 文件的结构，常用的 MIDI 指令，以及开发编辑器的技巧。

关键词：MIDI 文件；编辑器；MIDI 指令

1 引言

MIDI 文件是一种电子音乐文件。与常见的 WAV 文件不同，MIDI 文件里并不存储实际的声音数据，而是存储一系列 MIDI 指令。如果声卡支持 MIDI，就可以将 MIDI 指令发送给声卡，并驱动声卡发出声音。因此，MIDI 文件的数据量比 WAV 文件要小得多，同时 MIDI 具有丰富的音色库，可以表现出很多种乐器的声音。因此 MIDI 可以达到一个交响乐队的演奏效果。

下面将要开发一个 MIDI 编辑器。通过该程序，介绍一下 MIDI 文件的结构，以及常用的 MIDI 指令。MIDI 编辑器和文本编辑器有相似性，都是要实现符号的编辑功能，所不同的是，MIDI 编辑器所能输入的字符都是专用的音乐符号。MIDI 编辑器可以类似简谱的方式对 MIDI 文件中的数据进行编辑。

2 MIDI 文件结构

MIDI 文件由头块和轨道块两部分组成。头块存储一些全局性的信息。轨道块则存储每个轨道具体的 MIDI 指令。

头块的结构如表 1 所示。

表 1 头块的结构

字段名	长度	说明
id	4	值为 MThd，表明这是 MIDI 头块
size	4	值为 0 0 0 6，表明头块占 6 个字节
type	2	0 0-单音轨 0 1-同步多音轨 0 2-不同步多音轨
trackcount	2	该值表示 MIDI 文件中音轨个数
yf4ticks	2	该值表示每个四分音符所占的 MIDI tick 数。MIDI tick 是 MIDI 文件中所用的时间单位

轨道块结构如表 2 所示。

表 2 轨道块结构

字段名	长度	说明
id	4	值为 MTrk，表明这是 MIDI 轨道块
size	4	该值表示本轨道块字节数

轨道块中，size 之后为 MIDI 指令。常用的 MIDI 指令如表 3 所示。

表 3 常用的 MIDI 指令

MIDI 指令	说明
00 ff 51 03 07 a1 20	设置每个四分音符微秒数。第一个字节 0 表示距离上个 MIDI 指令时间差为 0；07 a1 20 三个字节换算为十进制数为 500000，即每个四分音符占用 500000 微秒
00 ff 58 04 04 02 18 08	设置节拍。第一个字节 0 表示距离上个 MIDI 指令时间差为 0；第二个 04 表示每小节有 4 拍；02 表示以 22,即以四分音符为 1 拍；18 表示节拍器 tick 数；08 表示每个四分音符包含 8 个三十二分音符
00 ff 59 02 00 00	设置调号。第一个字节 0 表示距离上个 MIDI 指令时间差为 0；倒数第二个 0 表示为 C 调，升号数目写成 0x，降号数目写成 8x，x 为升降调的数目；最后一个 0 表示大小调 0-大调 1-小调
00 bx 00 00 00 cx 00 00 bx 07 7f	其中 x 是通道号，从 0-f 共有 16 种选择。第一个 bx 之后两个 0 表示选择音色库；cx 之后那个字节表示选择的乐器编号，值从 0-127，共有 128 种不同的音色可供选择；第二个 bx 之后的 07 表示设置音量，7f 表示音量数值，数值范围从 0-127
tt 9x nn ss	该指令用于输出音符。tt 是距离上个 MIDI 指令的时间差；x 是通道号，从 0-f。nn 是输出的音符编码，范围从 0-127；ss 是按键力度，范围从 0-127，0 表示声音最小，127 表示声音最大。如果连续输出音符，每个 MIDI 指令的 9x 可以省略。
00 ff 2f 00	设置音轨结束。每个音轨末尾有该指令。

时间差在 MIDI 文件中比较重要。它表明两个相邻的 MIDI 指令间相距多少个 MIDI tick。如果 0<=时间差<=127，则用一个字节表示即可。如果 127<时间差<=16383，则用两个字节，第一个字节最高位为 1，剩余 7 位 *128+第二个字节，最后得出的数值即为时间差。如果 16383<时间差<=2097151，则用 3 个字节，前两个字节最高为 1，第一个字节剩余 7 位 *16384+第二个字节剩余 7 位 *128+第三个字节，最后得出的数值即为时间差。这种表示数值的方法叫做动态字节表示法。比如时间差为 60，则可以直接用 1 个字节 3c 表示。再比如时间差为 240，则可以将 240 拆分为 240=1*128+112，用 2 个字节表示为 81 70。再比如时间差为 65535，则可以将 65535 拆分为 65535=3*16384+127*128+127，用 3 个字节表示为 83 ff 7f。

3 编辑器开发技巧

和文本编辑器类似，MIDI 编辑器同样需要实现字符输入、删除、光标移动等基本功能。但和文本编辑器不同，MIDI 编辑器里字符主要是音符，因此在显示和输入方面有所区别。

MIDI 编辑器里主要音符有休止符、do、ri、mi、fa、so、la、xi 8 个音符，分别用数字 0、1、2、3、4、5、6、7 表示。这是中音部，如果提高一个 8 度，音符上方加 1 个圆点，如果再提高一个 8 度，音符上方加 2 个圆点。同理，降低一个 8 度，音符下方加 1 个圆点，如果再降低一个 8 度，音符下方加 2 个圆点。

对于音符的长度，可以这样表示。4 分音符直接用数字表示；8 分音符音符下方加一条横线；16 分音符音符下方加 2 条横线；32 分音符，音符下方加 3 条横线；二分音符，数字后加一个横线；5/8 全音符，数字后加一个圆点，再加一个该数字；3/4 全音符，数字后加 2 个横线；全音符，数字后加 3 个横线。

音轨号，用 [音轨 x] 表示，x 为音轨编号，范围从 0–15。

乐器编号，用 (乐器号 乐器名) 表示。

用一个字符串存储整个乐谱。其中，每个音符用两个字节表示，第一个字节表示音符的高低，第二个字节表示音符的长短。

使用方向键上、下、左、右来移动光标。

4 关键代码

读取 MIDI 文件，使用代码如下：

```
void CMidieditDlg::OnOpen()  //打开 MIDI 文件
{
    // TODO: Add your command handler code here
  CFileDialog *dlg;
  char filter[]="MIDI Files(*.mid)|*.mid||",s[100];
  CString str,str1,str2,str3;
  FILE *fp;
  unsigned char buf[1000],ch,i,ch1,ch2;
  unsigned short n;
  unsigned long n1;
  int r,t,j,ticks4yf,totallen,trackcount,fz,fm;
  HPEN pen1;
  HBRUSH brush1;

  //建立画笔和刷子
  pen1=CreatePen(PS_SOLID,1,RGB(236,233,216));
  brush1=CreateSolidBrush(RGB(236,233,216));
  SelectObject(hdc,pen1);
  SelectObject(hdc,brush1);
                    dlg=new CFileDialog(TRUE,"*.mid",NULL,
OFN_OVERWRITEPROMPT,
                                        filter,NULL);  //打开文件对话框
dlg->DoModal();
str=dlg->GetPathName();
if(str!  ="")
{
  m_static1.SetWindowText(str);  //显示文件名
    fp=fopen(str,"rb");
    if(fp!  =NULL)
    {
  row=1;
    line=0;
    *tracks="";
    cursorx=20;cursory=160;
    fseek(fp,8,SEEK_SET);
  fread(buf,1,6,fp);
    //显示类型
    if(buf[0]==0&&buf[1]==0)
      m_static2.SetWindowText("单音轨");
    if(buf[0]==0&&buf[1]==1)
      m_static2.SetWindowText("同步多音轨");
    if(buf[0]==0&&buf[1]==2)
      m_static2.SetWindowText("不同步多音轨");
    n=256*buf[2]+buf[3];
    trackcount=n;
    sprintf(s,"%u",n);
          m_static3.SetWindowText(s);  //显示音轨数
          n=256*buf[4]+buf[5];
          ticks4yf=n;
          sprintf(s,"%u",n);
            m_static4.SetWindowText(s);
  //显示 4 分音符 TICKS 数
          //显示 4 分音符微秒数
          fseek(fp,14,SEEK_SET);
          while(1)
          {
              r=fread(&ch,1,1,fp);
              if(r!  =1) break;
              if(ch==0)
              {
                r=fread(&ch,1,1,fp);
                if(r!  =1) break;
                if(ch==0xff)
  {
                  r=fread(&ch,1,1,fp);
                  if(r!  =1) break;
                  if(ch==0x51)
                  {
                    r=fread(&ch,1,1,fp);
                    if(r!  =1) break;
                    if(ch==3)
  {
                        fread(buf,1,3,fp);
```

```
                n1=65536*buf[0]+256*buf[1]+buf[2];
                sprintf(s,"%lu",n1);
                m_static5.SetWindowText(s);
                break;
            }
        }
    }
}
//显示节拍
fseek(fp,14,SEEK_SET);
while(1)
{
r=fread(&ch,1,1,fp);
    if(r! =1) break;
    if(ch==0)
    {
      r=fread(&ch,1,1,fp);
      if(r! =1) break;
 if(ch==0xff)
      {
          r=fread(&ch,1,1,fp);
          if(r! =1) break;
          if(ch==0x58)
          {
            r=fread(&ch,1,1,fp);
          if(r! =1) break;
          if(ch==4)
          {
              fread(buf,1,4,fp);
            ch=1;
            for(i=1;i<=buf[1];i++) ch=ch*2;
            sprintf(s,"%u/%u",buf[0],ch);
              fz=buf[0];
              fm=ch;
              m_static6.SetWindowText(s);
              break;
          }
        }
    }
  }
}
  //显示调号
  fseek(fp,14,SEEK_SET);
  strcpy(s,"C 大调");
while(1)
  {
      r=fread(&ch,1,1,fp);
      if(r! =1) break;
      if(ch==0)
      {
        r=fread(&ch,1,1,fp);
if(r! =1) break;
if(ch==0xff)
{
    r=fread(&ch,1,1,fp);
    if(r! =1) break;
    if(ch==0x59)
    {
      r=fread(&ch,1,1,fp);
      if(r! =1) break;
      if(ch==2)
      {
          fread(buf,1,2,fp);
          strcpy(s,"");
          if (buf [0] ==0)  strcat (s,"C");if (buf [0] ==1)
strcat(s,"bD");
          if (buf [0] ==2)  strcat (s,"D");if (buf [0] ==3)
strcat(s,"bE");
          if (buf [0] ==4)  strcat (s,"E");if (buf [0] ==5)
strcat(s,"F");
          if (buf [0] ==6)  strcat (s,"#F");if (buf [0] ==7)
strcat(s,"G");
          if(buf[0]==0x81) strcat(s,"B");if(buf[0]==
0x82) strcat(s,"bB");
          if(buf[0]==0x83) strcat(s,"A");if(buf[0]==
0x84) strcat(s,"bA");
              if(buf[1]==0) strcat(s,"大调");if (buf [1]
==1) strcat(s,"小调");
              m_static7.SetWindowText(s);
              break;
          }
        }
      }
    }
    //显示乐谱
    fseek(fp,14,SEEK_SET);
    fread(buf,1,8,fp);
    totallen=0;
    j=0;
  sprintf(s,"[音轨%u]\n",j);
    str=s;
    while(1)
  {
    r=fread(&ch,1,1,fp);
    if(r! =1) break;
    if(ch==0xff)
    {
      r=fread(&ch,1,1,fp);
      if(r! =1) break;
      if(ch==0x51)
      {
          r=fread(buf,1,4,fp);
```

```
    if(r! =4) break;
  }
  else
    if(ch==0x58)
      {
r=fread(buf,1,5,fp);if(r! =5) break;
      }
    else
      if(ch==0x59)
        {
          r=fread(buf,1,3,fp);if(r! =3) break;
        }
        else
          fseek(fp,-1,SEEK_CUR);
  }
  if(ch>>4==0xc) //设置乐器
  {
 fseek(fp,-5,SEEK_CUR);
    fread(buf,1,5,fp);
    if(! (buf[0]>>4==0xb&&buf[1]==0&&buf[2]==0))
continue;
    r=fread(&ch,1,1,fp);
    if(r! =1) break;
    if(ch==0)
      str1="(0 Acoustic Grand Piano 大钢琴 (声学钢
琴))\n";
  //省略其他乐器代码
      str=str+str1;
  }
  if(ch>>4==9) //开始音符输出
  {
    while(1)
    {
 r=fread(buf,1,2,fp);
      if(r! =2) break;
        if(buf[0]==0xff&&buf[1]==0x2f)
        {
          r=fread(&ch,1,1,fp);
          if(r! =1) break;
          if(ch==0)
          {
              r=fread(buf,1,8,fp);
        if(buf[0]==0x4d&&buf[1]==0x54&&
          buf[2]==0x72&&buf[3]==0x6b)
        {
          j++;
sprintf(s,"[音轨%u]\n",j);
          str=str+"\n";
  str=str+s;
          totallen=0;
          break;
        }
          }
          if(buf[1]==0)
          {
           if(buf[0]==0x3c) str1="0";
          }
          else
          {
if(buf[0]==0x3c) str1="1";if(buf[0]==0x3e) str1="2";
 if(buf[0]==0x40) str1="3";if(buf[0]==0x41) str1="4";
      if(buf[0]==0x43) str1="5";if(buf[0]==0x45) str1="6";
      if(buf[0]==0x47) str1="7";if(buf[0]==0x48) str1="h";
      if(buf[0]==0x4a) str1="i";if(buf[0]==0x4c) str1="j";
     if(buf[0]==0x4d) str1="k";if(buf[0]==0x4f) str1="l";
 if(buf[0]==0x51) str1="m";if(buf[0]==0x53) str1="n";
      if(buf[0]==0x54) str1="H";if(buf[0]==0x56) str1="I";
     if(buf[0]==0x58) str1="J";if(buf[0]==0x59) str1="K";
 if(buf[0]==0x5b) str1="L";if(buf[0]==0x5d) str1="M";
 if(buf[0]==0x5f) str1="N";if(buf[0]==0x30) str1="a";
 if(buf[0]==0x32) str1="b";if(buf[0]==0x34) str1="c";
 if(buf[0]==0x35) str1="d";if(buf[0]==0x37) str1="e";
if(buf[0]==0x39) str1="f";if(buf[0]==0x3b) str1="g";
 if(buf[0]==0x24) str1="A";if(buf[0]==0x26) str1="B";
 if(buf[0]==0x28) str1="C";if(buf[0]==0x29) str1="D";
 if(buf[0]==0x2b) str1="E";if(buf[0]==0x2d) str1="F";
          if(buf[0]==0x2f) str1="G";
          }
          r=fread(&ch1,1,1,fp);
          if(r! =1) break;
          if(ch1<=127)
        t=ch1;
          else
          {
            r=fread(&ch2,1,1,fp);
            if(r! =1) break;
            t=(ch1&0x7f)*128+ch2;
          }
          if(t==ticks4yf)
          {
            str2="1";totallen=totallen+8;
          }
          if(t==ticks4yf/2)
          {
            str2="2";totallen=totallen+4;
          }
        if(t==ticks4yf/4)
          {
            str2="3";totallen=totallen+2;
          }
        if(t==ticks4yf/8)
          {
            str2="4";totallen=totallen+1;
```

```
    }
  if(t==ticks4yf*2)
    {
      str2="5";totallen=totallen+16;
    }
  if(t==ticks4yf*4)
    {
      str2="6";totallen=totallen+32;
    }
  if(t*2==ticks4yf*5)
    {
      str2="7";totallen=totallen+20;
    }
  if(t==ticks4yf*3)
    {
      str2="8";totallen=totallen+24;
    }
  r=fread(buf,1,2,fp);
  if(r! =2) break;
    r=fread(&ch,1,1,fp);
    if(r! =1) break;
  str=str+str1+str2;
    if(totallen>=128&&totallen%(32*fz/fm)==0)
    {
      str=str+"\n";totallen=0;
    }
                }
              }
            }
        }
        str=str+"\n";
        str1=str.Right(2);
        if(str1=="\n\n") str.Delete(str.GetLength()-1);
        *tracks=str;
      AfxGetMainWnd()->HideCaret();
        Rectangle(hdc,20,150,860,630);
      ShowTracks(line);
        fclose(fp);
      }
  }
  DeleteObject(pen1);DeleteObject(brush1);
}
```

保存 MIDI 文件，使用代码如下：

```
void CMidieditDlg::OnSave()  //保存 MIDI 文件
{
      // TODO: Add your command handler code here
  CString str,str1,str2;
  FILE *fp;
  unsigned char buf[1000],ch,s1[100];
  unsigned long len,pos;
  int n,n1,n2,trackcount,i,j,startpos,length,t,r,ticks4yf,flag;
  char s[1000];
            AfxGetMainWnd ()->GetDlgItem (IDC_STATIC1)->
GetWindowText(str);
  fp=fopen(str,"wb");
  if(fp! =NULL)
  {
    //写入文件头
    buf[0]=0x4d;buf[1]=0x54;buf[2]=0x68;buf[3]=0x64;
    fwrite(buf,1,4,fp);
    buf[0]=0x00;buf[1]=0x00;buf[2]=0x00;buf[3]=0x06;
    fwrite(buf,1,4,fp);
          AfxGetMainWnd () ->GetDlgItem (IDC_STATIC2) ->
GetWindowText(str);
    buf[0]=0;buf[1]=0;
      if(str=="单音轨")
      {
        buf[0]=0;buf[1]=0;
      }
      if(str=="同步多音轨")
      {
        buf[0]=0;buf[1]=1;
      }
      if(str=="不同步多音轨")
      {
        buf[0]=0;buf[1]=2;
      }
      fwrite(buf,1,2,fp);
    buf[0]=0;buf[1]=1;
          AfxGetMainWnd () ->GetDlgItem (IDC_STATIC3) ->
GetWindowText(str);
      if(str! ="")
      {
        n=atoi(str);
        trackcount=n;
        buf[0]=(unsigned char)(n/256);buf[1]=n%256;
      }
      fwrite(buf,1,2,fp);
      buf[0]=1;buf[1]=0xe0;
      ticks4yf=480;
          AfxGetMainWnd () ->GetDlgItem (IDC_STATIC4) ->
GetWindowText(str);
      if(str! ="")
      {
        n=atoi(str);
        ticks4yf=n;
        buf[0]=(unsigned char)(n/256);buf[1]=n%256;
      }
      fwrite(buf,1,2,fp);
      for(i=1;i<=trackcount;i++)  //写入音轨
      {
        buf[0]=0x4d;buf[1]=0x54;buf[2]=0x72;buf[3]=0x6b;
        fwrite(buf,1,4,fp);
        pos=ftell(fp);
```

```
len=0;
buf[0]=0;buf[1]=0;buf[2]=0;buf[3]=0;
fwrite(buf,1,4,fp);
buf[0]=0;buf[1]=0xff;buf[2]=0x51;buf[3]=0x03;
buf[4]=0x07;buf[5]=0xa1;buf[6]=0x20;
AfxGetMainWnd () ->GetDlgItem (IDC_STATIC5) ->
GetWindowText(str);
if(str! ="")
{
n=atoi(str);
buf[4]=(unsigned char)(n/65536);
buf[5]=(unsigned char)((n/256)&0x00ff);
buf[6]=n%256;
}
fwrite(buf,1,7,fp); //写入四分音符微秒数
len=len+7;
buf[0]=0;buf[1]=0xff;buf[2]=0x58;buf[3]=0x04;
buf[4]=0x04;buf[5]=0x02;buf[6]=0x18;buf[7]=0x08;
AfxGetMainWnd () ->GetDlgItem (IDC_STATIC6) ->
GetWindowText(str);
if(str! ="")
{
str1=str;str1=str1.Mid(0,str1.Find('/'));n1=atoi(str1);
str1=str;str1=str1.Mid(str1.Find('/')+1);n2=atoi(str1);
buf[4]=(unsigned char)n1;
if(n2==1) buf[5]=0;if(n2==2) buf[5]=1;
if(n2==4) buf[5]=2;if(n2==8) buf[5]=3;
}
fwrite(buf,1,8,fp); //写入节拍
len=len+8;
buf[0]=0;buf[1]=0xff;buf[2]=0x59;buf[3]=0x02;
buf[4]=0x00;buf[5]=0x00;
AfxGetMainWnd () ->GetDlgItem (IDC_STATIC7) ->
GetWindowText(str);
if(str! ="")
{
if(str[0]=='C') buf[4]=0;
if(str[0]=='b'&&str[1]=='D') buf[4]=1;
if(str[0]=='D') buf[4]=2;
if(str[0]=='b'&&str[1]=='E') buf[4]=3;
if(str[0]=='E') buf[4]=4;
if(str[0]=='F') buf[4]=5;
if(str[0]=='#'&&str[1]=='F') buf[4]=6;
if(str[0]=='G') buf[4]=7;
if(str[0]=='B') buf[4]=0x81;
if(str[0]=='b'&&str[1]=='B') buf[4]=0x82;
if(str[0]=='A') buf[4]=0x83;
if(str[0]=='b'&&str[1]=='A') buf[4]=0x84;
if(str.Find("大调")! =-1) buf[5]=0;
if(str.Find("小调")! =-1) buf[5]=1;
}
fwrite(buf,1,6,fp); //写入调号
len=len+6;
str=*tracks;
startpos=0;
while(1)
{
str1=str.Mid(0,str.Find('\n'));
startpos=startpos+str1.GetLength()+1;
if(str1.Find('[')! =-1)
{
str2=str1.Mid(5,str1.Find(']'));
if(atoi(str2)==i-1)
{
str=*tracks;
str=str.Mid(startpos);
j=0;
while(1)
{
if(str[j]=='[')
{
length=j;break;
}
j++;
if(j+startpos>tracks->GetLength())
{
length=j-1;break;
}
}
break;
}
}
str=str.Mid(str.Find('\n')+1);
if(str=="") break;
}
str=*tracks;
str=str.Mid(startpos,length);
while(1)
{
str=str.Mid(0,str.Find('\n'));
if(str.Find('(')! =-1)
{
str1=str.Mid(1,str.Find(' '));
ch=(unsigned char)atoi(str1);
buf[0]=0;
buf[1]=0xb0+(unsigned char)i-1; //选择音色库
buf[2]=0;buf[3]=0;
buf[4]=0;
buf[5]=0xc0+(unsigned char)i-1; //改变乐器
buf[6]=ch;
buf[7]=0;
buf[8]=0xb0+(unsigned char)i-1; //设置音量
buf[9]=7;buf[10]=0x7f;
fwrite(buf,1,11,fp); //写入乐器选择
```

```
        len=len+11;
      }
      if(str=="") break;
}
buf[0]=0;buf[1]=0x90+(unsigned char)i-1;
fwrite(buf,1,2,fp);
len=len+2;
str=*tracks;
str=str.Mid(startpos,length);
flag=0;
while(1)
{
    str1=str.Mid(0,str.Find('\n'));
    if(str1=="") break;
 if(str1.Find('(')==-1)
    {
 strcpy(s,str1);
      for(j=1;j<=(int)strlen(s);j+=2)
      {
        if(flag==0)
    flag=1;
           else
            if(flag==1)
            {
            buf[0]=0;
              fwrite(buf,1,1,fp);
            len=len+1;
          }
if(s[j-1]=='1') ch=0x3c;if(s[j-1]=='2') ch=0x3e;
if(s[j-1]=='3') ch=0x40;if(s[j-1]=='4') ch=0x41;
if(s[j-1]=='5') ch=0x43;if(s[j-1]=='6') ch=0x45;
if(s[j-1]=='7') ch=0x47;if(s[j-1]=='h') ch=0x48;
      if(s[j-1]=='i') ch=0x4a;if(s[j-1]=='j') ch=0x4c;
if(s[j-1]=='k') ch=0x4d;if(s[j-1]=='l') ch=0x4f;
if(s[j-1]=='m') ch=0x51;if(s[j-1]=='n') ch=0x53;
if(s[j-1]=='H') ch=0x54;if(s[j-1]=='I') ch=0x56;
if(s[j-1]=='J') ch=0x58;if(s[j-1]=='K') ch=0x59;
if(s[j-1]=='L') ch=0x5b;if(s[j-1]=='M') ch=0x5d;
if(s[j-1]=='N') ch=0x5f;if(s[j-1]=='a') ch=0x30;
if(s[j-1]=='b') ch=0x32;if(s[j-1]=='c') ch=0x34;
if(s[j-1]=='d') ch=0x35;if(s[j-1]=='e') ch=0x37;
if(s[j-1]=='f') ch=0x39;if(s[j-1]=='g') ch=0x3b;
if(s[j-1]=='A') ch=0x24;if(s[j-1]=='B') ch=0x26;
if(s[j-1]=='C') ch=0x28;if(s[j-1]=='D') ch=0x29;
if(s[j-1]=='E') ch=0x2b;if(s[j-1]=='F') ch=0x2d;
      if(s[j-1]=='G') ch=0x2f;
      if(s[j-1]! ='0')
      {
        buf[0]=ch;buf[1]=0x40;
      }
      else
      {
        buf[0]=0x3c;buf[1]=0;
      }
    fwrite(buf,1,2,fp);  //写入音符
      len=len+2;
  if(s[j]=='1') t=ticks4yf;
  if(s[j]=='2') t=ticks4yf/2;
  if(s[j]=='3') t=ticks4yf/4;
  if(s[j]=='4') t=ticks4yf/8;
  if(s[j]=='5') t=ticks4yf*2;
  if(s[j]=='6') t=ticks4yf*4;
    if(s[j]=='7') t=ticks4yf*5/2;
    if(s[j]=='8') t=ticks4yf*3;
 r=numbertochar(t,s1);
        fwrite(s1,1,r,fp);
        len=len+r;
        if(s[j-1]! ='0')
        {
          buf[0]=ch;buf[1]=0;
        }
        else
        {
          buf[0]=0x3c;buf[1]=0;
        }
        fwrite(buf,1,2,fp);
        len=len+2;
      }
      }
      str=str.Mid(str.Find('\n')+1);
    }
    buf[0]=0;buf[1]=0xff;buf[2]=0x2f;buf[3]=0x00;
    len=len+4;
    fwrite(buf,1,4,fp);  //写入音轨结束符
    fseek(fp,pos,SEEK_SET);
    buf[0]=(unsigned char)(len/16777216);
    buf[1]=(unsigned char)(len/65536);
    buf[2]=(unsigned char)(len/256);
 buf[3]=(unsigned char)(len%256);
    fwrite(buf,1,4,fp);  //修改音轨长度
    fseek(fp,0,SEEK_END);
  }
  fclose(fp);
  AfxMessageBox("保存成功");
 }
}
```

5 结论

本程序开发完成，实现了 MIDI 编辑器的基本功能。可以在此基础上进行完善，加入更多的实用功能。

（收稿日期：2012-12-11）

IE8 加速器开发小议

李 斌

摘 要：描述了 IE8 加速器的定义、配置、安装等开发的注意事项，同时给出了若干具体示例。

关键词：IE8 浏览器；加速器

1 引言

随着 Internet Explore 8 的发行，此款浏览器的功能进行了大幅扩展，其中一个新增加的功能就是“加速器” (Accelerator)。所谓加速器，顾名思义就是加快用户访问网页的速度，如果读者使用过 IE8，就可看到类似图 1 的界面，其中的小方框和弹出式菜单就是加速器，如果点击菜单中的项目，就会打开一个新的网页，这个新网页使用了原来网页中的某些信息作为输入，例如利用原来网页中复制的文字作为新打开的搜索页面的待搜索项，值得注意的是，当在原来网页中选择某个文本时，小方框就会自动出现，同时点击菜单后，选择的文本就会自动被传送给搜索页面，这比利用 CTRL+C 复制再到搜索页去 CTRL+V 粘贴到搜索输入框后点击搜索按钮要快捷得多，这就是“加速器”加速的含义所在。

图 1 IE8 中的加速器

2 加速器介绍

下面详细说明一下加速器的定义。所谓加速器是指一些可从任何网页快速访问应用程序或 Web 服务的上下文菜单选项。如图 1 所示，用户通过选择一些文本后触发了上下文菜单，用户通过点击菜单达到访问其他网页程序或服务的目标。这里必须指出，加速器还有一个“预览”功能非常实用，所谓“预览”就是当用户将鼠标指针移动到上下文菜单某项时，如果该加速器定义了预览功能，那么浏览器会立即弹出一个小窗体(320*240 像素) 来显示要访问的网站页面，这样用户可以达到不转向其他页面就获得服务的目标，无疑更为便捷，当然预览窗口被强制定义为 320*240 像素，如果加速器访问的页面超过这个大小，浏览器会自动进行剪裁，这就对预览页面提出了更高的要求，后文介绍示例时会做说明。

为了提高用户访问加速器的速度，IE 浏览器对加速器实行分组显示，因此加速器需要在定义中指示其自身类别，有一些现成的分类，例如 IE8 默认安装后会生成一个 Windows Live Writer 的加速器，此加速器用于撰写 Window Live 博客，因此该加速器属于“博客”组，又比如读者可以安装 Google Map 加速器来访问 Google 地图服务，显然这个加速器属于“地图”组。用户还可自行添加自定义分组，例如笔者后文的例子中就添加了 Search Baidubaike 组，每个分组可由用户指定一个默认加速器，这样该加速器将显示在上下文菜单的显要位置，如图 1 所示。

加速器通过网站来安装，网站上部署一个加速器定义文件，文件用 XML 格式描述，用户通过浏览器打开此 XML 文件来安装加速器，下面给出一个加速器的 XML 定义文件，并结合此文件来介绍一下加速器的定义文件的编写方法：

```
<?xml version="1.0" encoding="UTF-8"?>
    <os:openServiceDescription xmlns:os = "http://www.microsoft.com/schemas/openservicedescription/1.0">
      <os:homepageUrl >http://maps.example.com </os:homepageUrl>
  <os:display>
    <os:name>Map with MyMap</os:name>
     <os:icon >http://www.example.com/favicon.ico </os:icon>
    <os:description>Map addresses easily with MyMap.</os:description>
  </os:display>
  <os:activity category="Map">
    <os:activityAction context="selection">
      <os:preview action = "http://maps.example.com/preview.php?addr={selection}" />
      <os:execute action = "http://maps.example.com/" method="get">
        <os:parameter name = "addr" value = "{selection}" type="text" />
      </os:execute>
```

```
    </os:activityAction>
  </os:activity>
</os:openServiceDescription>
```

定义文件的第一行是一个 XML 文件的文件头,第 2 行是固定的，定义了 openServiceDescription 节点，下面是 homepage 节点，表明加速器的主页地址，请注意，这个节点定义的主页位置需要与后面定义具体加速器预览或激活地址位于一个域名之下，否则加速器可能无法正常工作。下面是 display 节点，这个节点下的 name,icon,description 子节点定义了上下文菜单中的显示项。下面是最重要的 activity 节点，这个节点定义了加速器的具体行为，首先这里指明了要定义加速器的分类，即 category 项，可以用预定义的分类，也可以自定义分类，随后子节点 activityAction 具体定义加速器究竟激活哪个网络服务地址，这里的 context 项很重要，它定义了何种情况下上下文菜单会弹出，示例中使用 selection 表明当用户在浏览器中选择文本时会激活菜单，context 可设置的内容如表 1 所示。

表 1　context 项设置

Context	说明
document	当前文档。 始终可用。
selection	默认设置。 选定的文本。 加速器仅在单击选定区域时可用。
link	超链接。 加速器仅对链接可用。

随后 activityAction 的两个子节点 preview 和 execute 分别具体定义了用户预览或点击加速器时的具体行为，在这两个节点中可以设置以下项，具体如表 2 所示。

表 2　activityAction 的子节点设置

属性	是否必需?	说明
action	是	用于 HTTP 提交的 URL 模板。
method	否	要使用的 HTTP 方法的类型（get、post）。 默认值为 get。
enctype	否	提交到服务器的内容的类型。 默认值为 application/x-www-form-urlencoded。
accept-charset	否	用于提交的字符集。 默认值为 utf-8。

这里 action 是必须提供的项，它表明了加速器要访问的 http 地址，method 项标明了 HTTP 方法的类型是 Get 还是 Post，这里特别注意，不管是 Get 还是 Post 的 HTTP 访问，都可能带有参数，这些参数既可以写在 action 项内，如果参数较复杂，则可以设置 preview 或 execute 节点的子节点 parameter 来定义参数，该节点可设置 name、value、type 3 个属性，其中 name 应该对应于需访问 http 页面的参数名，而 value 是要传送参数的值，这个值通常会引用加速器变量，这里简单说明一下何为加速器变量，弹出加速器上下文菜单的页面中的一些值可以作为要访问的加速器 http 地址页面的输入参数，那么这些值就成为了加速器变量，根据于 activityAction 节点的 Context 项的取值，可选的加速器变量不尽相同，具体如表 3 所示。

表 3　Context 项的取值和加速器变量的选择

变量	Context	说明
{documentUrl}	全部	文档的 href。
{documentTitle}	全部	文档的 title（如果有）。
{documentDomain}	全部	文档的 href 中的有效二级域。
{documentHost}	全部	文档的 href 中的完全限定域。
{selection}	selection	当前选定的文本。
{link}	link	选定链接的 href。
{linkText}	link	选定链接的 innerText。
{linkRel}	link	选定链接的 rel（如果可用）。
{linkType}	link	选定链接的 type（如果可用）。
{linkDomain}	link	链接的 href 中的有效二级域。
{linkHost}	link	链接的 href 中的完全限定域。

比较常用的就是当 Context 取 selection 值时传递 {selection} 变量给 http 地址，下面的示例也采用了这种传递方式。这里还有一点需要指出，上面介绍 action 节点和 preview 节点时，没有说明 accept-charset 属性，事实上这个属性对于经常访问中文页面的我们很重要，虽然现在许多页面已经使用 UTF-8 来编码，但是还是有许多中文网页采用了 GB2312 编码，此时传递参数时必须手工指定 accept-charset 属性，下文的示例将展示这个问题。

至此，一个简单的加速器定义文件已经编写完毕，再使用 javascript 语言来操纵浏览器实施安装并创建一个简单的 html 网页即可，例如下面的示例代码：

```
<html>
<head>
<title> setup accelerator </title>
</head>
<body>
        <button  id = "myButton"  onclick = "window.external.
AddService('http://www.example.com/activity.xml')"> Add
 MyAccelerator to the shortcut menu in Internet Explorer 8</
button>
</body>
```

```
</html>
```

事实上，JavaScript 语言通过调用 AddService 方法来进行加速器安装，该方法所带参数就是刚才详细说明的加速器定义文件的 http 地址。

3 实例说明

下面给出两个具体例子，如图 1 所示，笔者建立了一个访问百度百科和访问 mtime 网搜索电影的加速器，以访问百度百科为例，原来要查询某个词条，必须先复制词条名，然后打开百度百科主页，将词条名粘贴到网页搜索框，然后点击“进入百科”按钮，这样，该词条的百科页面才能显示出来，创建加速器后，只须在网页上对感兴趣的词条进行复制，随后点击自动出现的上下文菜单，百科页面就能显示出来，以下是这个加速器的定义文件：

```
<?xml version="1.0" encoding="UTF-8"?>
<os:openServiceDescription
        xmlns:os = "http://www.microsoft.com/schemas/openservicedescription/1.0">
        <os:homepageUrl >http://baike.baidu.com/</os:homepageUrl>
    <os:display>
        <os:name>Search Baidu Baike</os:name>
        <os:description >Search Content in Baidu Baike </os:description>
    </os:display>
    <os:activity category="Searchbaidubaike">
        <os:activityAction context="selection">
            <os:preview action = "http://baike.baidu.com/searchword/" method="get" accept-charset="gb2312" >
            <os:parameter name = "word" value = "{selection}" type="text" />
        <os:parameter name="pic" value="1" type="text" />
         <os:parameter name="sug" value="1" type="text" />
           </os:preview>
            <os:execute action = "http://baike.baidu.com/searchword/" method="get" accept-charset="gb2312" >
            <os:parameter name = "word" value = "{selection}" type="text" />
        <os:parameter name="pic" value="1" type="text" />
          <os:parameter name="sug" value="1" type="text" />
            </os:execute>
        </os:activityAction>
    </os:activity>
</os:openServiceDescription>
```

这里大部分内容前文已说明过，只特别指出两点，首先 action 属性设置的 http 地址的获取需要使用一些 http 分析工具，例如 httpwatch，限于篇幅，这里不做详细展开，读者可参考其他资料或笔者之前发表的《利用 Google 打造自己的词霸和快译》一文，其次，baidu 网采用了 GB2312 编码，因此上面定义文件中设置了 accept-charset 属性。下面再给出访问 mtime 网的加速器定义文件，通过此加速器可方便地切换到感兴趣的电影信息页面，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<os:openServiceDescription
        xmlns:os = "http://www.microsoft.com/schemas/openservicedescription/1.0">
        <os:homepageUrl >http://search.mtime.com/</os:homepageUrl>
    <os:display>
        <os:name>Search Mtime</os:name>
        <os:description >Search Content in mtime </os:description>
    </os:display>
    <os:activity category="Searchmtime">
        <os:activityAction context="selection">
            <os:preview action = "http://search.mtime.com/search/?{selection}" />
            <os:execute action = "http://search.mtime.com/search/?{selection}" />
        </os:activityAction>
    </os:activity>
</os:openServiceDescription>
```

这里只说明一点，对于参数简单的传递，可以直接在 action 属性中指定参数而无须再额外定义 parameter 节点了。

最后再强调一点，加速器的 preview 功能只提供 320*240 大小的窗口，上面的例子中 preview 页面与 action 页面是一致的，这种情况下 preview 的窗口可能由于剪裁而无法提供有效的信息，读者可自行重新设计 preview 访问的 http 服务来解决这个问题，当然如何根据 preview 窗口大小来设计更高效的页面已经超出了文中的范围，读者可自行研究。

4 结语

介绍了 IE8 的新功能——加速器的基本开发方法，借此文抛砖引玉，读者可进一步研究，用好 IE8 的这个新工具，开发出更多加速网页访问的利器，为更好地畅游网络不懈努力添砖加瓦。

参考文献

[1] OpenService Accelerators Developer Guide.

[2] http://msdn.microsoft.com/zh-cn/library/cc289775 (en-us,VS.85).aspx.

（收稿日期：2012-12-12）

软件“忙状态”信息显示面板设计及应用

刘仁轩

摘　要：给出基于VC的软件运行时“忙状态”信息显示面板的动态链接库实现，并给出该链接库在VB.Net下的调用示例。

关键词：VC语言；忙状态；信息显示；动态链接库

1　引言

在软件设计过程中，经常会遇到当软件某个功能进入工作状态后，由于其完成需要较长时间，这时软件进入“忙状态”，给用户的感觉是软件此时无响应，键盘和鼠标等外部设备也停止了工作。对于这种情况，一般的解决方法有两种：方法一：新建一个工作线程，将该功能模块从主线程移到新建的工作线程中；方法二：通过特定界面，为用户提供相应的状态提示，使其明确此时软件正在干什么，从而避免认为软件“死机”。

针对以上情况，给出一个基于VC的动态链接库，当软件进入“忙状态”时，开发人员通过调用接口函数，在屏幕中央位置出现一个长条状的显示面板，告诉用户目前软件正在干什么，当“忙状态”结束，再通过接口函数关闭该面板，从而避免了用户的误解。

2　实现

2.1　设计思想

动态链接库InfoPanel采用VC实现。只提供一个stdcall调用的API接口SetInfoText，原型为：void SetInfoText (const char * lpszText)；参数lpszText为要在面板上显示的“忙”信息。调用流程如图1所示。

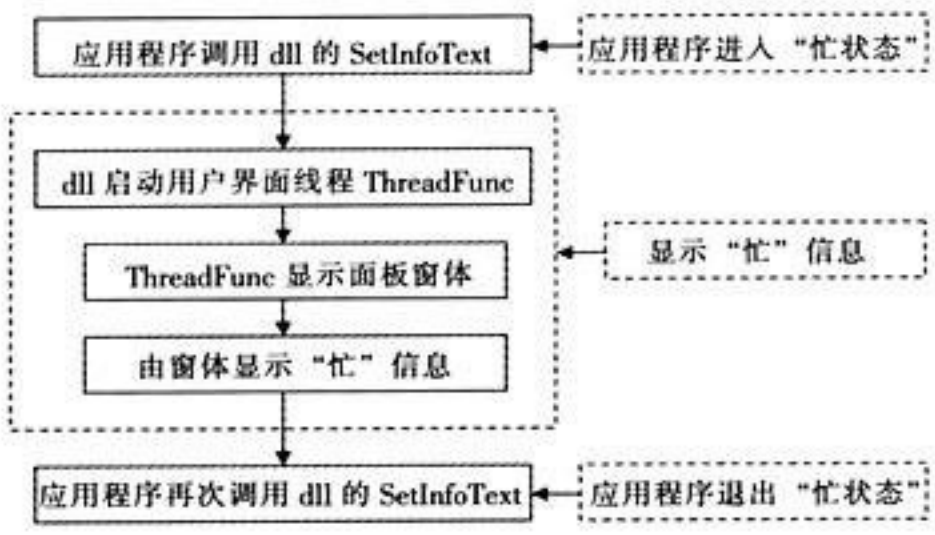


图1　动态链接库调用流程

当应用程序需要启动面板时，调用InfoPanel.dll的输出函数SetInfoText。这时dll将启动用户界面线程ThreadFunc，线程为用户显示面板窗体，并显示相应的“忙”信息。当应用程序“忙状态”结束，只需再次调用SetInfoText，并将其中的参数lpszText设置为空字符串，这时dll将关闭面板窗体，退出用户界面线程，一次完整的调用结束。应用程序可继续后续功能。

2.2　具体实现

```
//全局变量
HWND hWndMain=NULL; //面板窗体句柄
char szInfoText[512]; //“忙”信息文本
HANDLE hThread=NULL; //用户界面线程句柄
HINSTANCE hInst=NULL; //dll 实例句柄
HBRUSH hBrush; //用户面板背景画刷
BOOL APIENTRY DllMain ( HANDLE hModule, DWORD
ul_reason_for_call, LPVOID lpReserved ) //dll 入口函数
{
    hInst=(HINSTANCE)hModule;
                hBrush =CreateSolidBrush (GetSysColor
(COLOR_INFOBK)); //创建面板窗体背景画刷
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
LRESULT CALLBACK WndProc (HWND hWnd, UINT
message, WPARAM wParam, LPARAM lParam) //面板窗体
//消息处理函数
{
    PAINTSTRUCT ps;
    HDC hdc;
    RECT rc;
    HGDIOBJ hFont;
    switch (message)
    {
    case WM_CREATE:
        return 0;
    case WM_PAINT:
```

```
        //对于面板的绘制消息处理
        hdc = BeginPaint(hWnd, &ps);
    GetClientRect(hWnd, &rc);  //获得面板的客户区大小
    FillRect(hdc, &rc, hBrush);  //用背景画刷填充面板背景
        SetBkMode(ps.hdc, TRANSPARENT);
        SetTextColor(ps.hdc, RGB(0,0,0));
            hFont=SelectObject (ps.hdc, GetStockObject
(DEFAULT_GUI_FONT));
DrawText(ps.hdc, szInfoText, -1, &ps.rcPaint, DT_SINGLELIN
E | DT_VCENTER | DT_CENTER);  //绘制"忙"状态文本
            SelectObject(ps.hdc, hFont);
            EndPaint(hWnd, &ps);
            break;
        case WM_DESTROY:
            PostQuitMessage(0);
            break;
        default:
   return DefWindowProc(hWnd, message, wParam, lParam);
        }
        return 0;
}
DWORD WINAPI ThreadFunc(LPVOID param)  //用户工作线
//程主函数
{
        //创建面板窗体
        WNDCLASSEX wcex;
        wcex.cbSize = sizeof(WNDCLASSEX);
        wcex.style = CS_HREDRAW | CS_VREDRAW;
        wcex.lpfnWndProc = (WNDPROC)WndProc;
        wcex.cbClsExtra = 0;
        wcex.cbWndExtra = 0;
        wcex.hInstance = hInst;
        wcex.hIcon = NULL;
        wcex.hCursor = LoadCursor(NULL, IDC_WAIT);
//显示"等待"光标
        wcex.hbrBackground= hBrush;
        wcex.lpszMenuName = NULL;
        wcex.lpszClassName = "lrxinfopanel";
        wcex.hIconSm = NULL;
        RegisterClassEx(&wcex);
        HWND hWnd;
        int x,y,cx,cy;
        cx=600;cy=60;  //面板窗体的大小采用预定义大小,位置
//在屏幕正中央
        x=(GetSystemMetrics(SM_CXSCREEN)-cx)/2;
        y=(GetSystemMetrics(SM_CYSCREEN)-cy)/2;
hWnd = CreateWindowEx(WS_EX_NOACTIVATE| WS_EX_T
OPMOST,"lrxinfopanel","",WS_POPUP|WS_DLGFRAME| WS_
DISABLED, x, y, cx, cy, NULL, NULL, hInst, NULL);
        if (! hWnd)  return 0;
        hWndMain=hWnd;
ShowWindow(hWnd, SW_SHOW);//创建成功后显示面板窗体
        UpdateWindow(hWnd);
        //进入面板窗体消息循环
        MSG msg;
        while (GetMessage(&msg, NULL, 0, 0)! =0)
        {
            TranslateMessage(&msg);
            DispatchMessage(&msg);
        }
        DeleteObject(hBrush);
        return 0;
}
void SetInfoText(const char * lpszText)  //dll 输出函数
{
        if (lpszText==NULL || strlen(lpszText)==0)  //当调用参数
//为空时,终止线程
        {
            TerminateThread(hThread, 0);
            return;
        }
        if (strlen(lpszText)>sizeof(szInfoText))  //确保有效长度的
//"忙"信息文本
            strncpy(szInfoText, lpszText, sizeof(szInfoText));
        else
            strcpy(szInfoText, lpszText);
        //当线程没有退出时,表明应用程序正在上一个"忙状态"
//中,此时只需更新窗体文本即可
        DWORD dwCode;
        if(GetExitCodeThread(hThread, &dwCode)==TRUE)
        {
            if (dwCode==STILL_ACTIVE)
            {
                InvalidateRect(hWndMain,NULL, TRUE);
                return;
            }
        }
        //当所有有效验证通过后,创建用户界面线程
        DWORD dwThreadId;
            hThread = CreateThread( NULL,0,ThreadFunc,
0,0,&dwThreadId);
}
```

2.3 调用

这里给出在 VB.Net 中的调用示例。

声明如下:

```
    <System.Runtime.InteropServices.DllImportAttribute  ("
InfoPanel.dll")> _
  Public Sub SetInfoText(ByVal szText As String)
  End Sub
```

当程序进入“忙状态”后,显示面板时可如下调用:

```
SetInfoText("正在连接远程数据库……")
```

当“忙”状态结束后,关闭面板时如下调用:

```
Setinfotext("")
```

(下转第 34 页)

在 Delphi 中 DLL 的制作与应用

魏景东

摘　要：通过一个实例，介绍了在 Delphi7.0 中制作动态链接库 DLL 及 API 函数调用。

关键词：Delphi 语言；Windows DLL 库；Windows API 函数

1　引言

DLL（动态链接库）是一个用于其他执行模块调用的可执行模块，其扩展名为.DLL。该模块包含有可被其他应用程序和其他 DLL 共享的函数和资源。其最大特点是，其代码在应用程序运行期间被动态地链接。DLL 一旦装入内存，DLL 函数就可以被系统中任何正在运行的应用程序软件所应用而不必重复编译或链接。使用 DLL 具有好处：（1）DLL 达到了复用代码的极限，因直到程序运行并调用一个 DLL 函数时，该程序才要求给出这个函数的地址，此时 Windows 才在 DLL 中寻找被调用的函数并把该函数的地址传送给调用程序。（2）对 DLL 中函数的修改可自动传播到所有调用它的程序，而不必对程序做任何改动或处理。（3）提供了重用函数的机制和数据共享的机制。只包含共享数据的 DLL 称为资源文件，如 Windows 的字体文件等。

Windows 被大量的 DLL 支持，其中包括 Windows API 函数、各种驱动程序文件和各种带有.Fon 和.Fot 扩展名的字体资源文件。Delphi 把 Windows API 函数和其他 Windows DLL 函数重新组织到了几个库单元中，用户不必使用特殊的调用格式。下面通过一个试例介绍在 Delphi7.0 中如何制作动态链接库 DLL 及如何在应用软件中调用动态链接库 DLL 中的 API 函数。

2　DLL 的制作及其 API 函数调用

2.1　DLL 的制作

以下实例包含 3 个 API 函数、1 个字符串全局变量 jztime。3 个 API 函数的功能分别是求两个整数的最大值、最小值及把字符串全局变量 jztime 输出，字符串全局变量 jztime 在 DLL 加载内存时被赋值为 DLL 加载内存时间。

在对象库的 New 选项卡上，Delphi7.0 提供了一个 DLL 模块，利用该 DLL 模块，可以快速建立一个最简单的 DLL 工程。对该 DLL 工程文件进行修改并命名为 P_350 保存，完整代码如下：

```
library P_350;
uses
  SysUtils,
  System,
  windows,
  Classes;
var
   jzsj:string='0';  //定义 DLL 公用变量
{$R *.res}
function min(x,y:integer):integer;stdcall; //求最小值的 API 函数
   begin
     if x<y then min:=x else min:=y;
   end;
function max(x,y:integer):integer;stdcall;//求最大值的 API 函数
   begin
     if x>y then max:=x else max:=y;
   end;
function jztime():string;stdcall;//输出 DLL 公用变量值的 API 函数
   begin
     jztime:=jzsj;
   end;
exports  // 输出 DLL 的 API 函数
  min,jztime, max;
begin
     jzsj:=DateTimeToStr(Now);  //在 DLL 加载时对 DLL
//公用变量初始化
end.
```

然后在 Delphi7.0 开发工具顶菜单 Project 中点击Build，如果没错，在当前目录中生成 P_350.DLL 文件，该 P_350.DLL 就是制作的 DLL 实例动态链接库。

2.2　API 函数调用

调用一个储存在 DLL 中 API 函数有静态和动态两种方式。静态调用方式就是在单元的 Interface 部分用 External 指示字列出要从 DLL 调用的函数；动态调用方式是通过调用 Windows API 的 LoadLibrary（或 LoadLibraryEx）函数、GetProAddress 函数以及 FreeLibrary 函数动态地调用 DLL 中 API 函数。

使用静态调用方式所需的代码量很少，但有两个缺点：一是当要加载的 DLL 不存在或 DLL 中没有要调用的 API 函数时，程序就会自动停止运行；二是一旦 DLL 加载，就一直停留在应用程序的地址空间，即使 DLL 已经不再需要。

动态调用方式是在调用前引入，并且 LoadLibrary 函数可以指定不同的 DLL，GetProAddress 函数可以指定不同的 API 函数，如指定的 DLL 出错，最多是 API 函数动态失败，不会导

致程序终止。

在该试例中，使用 Delphi7.0 开发工具创建一个 Application 并命名为 P350DLL1，为简单起见，假设该应用仅有一个窗体 Form1，在该窗体上放置两个按钮 Button1、Button2，其 Caption 属性依次设置为：DLL 静态调用、DLL 动态调用。Form1 的 Caption 属性设置为：

动态链接库函数的两种调用方法。窗体 Form1 的设计示意如图 1 所示。

图 1 窗体 Form1 设计示意图

并将其保存为 P350dll.pas。

P350dll.pas 完整实现代码如下：

```
unit P350Dll;
interface
uses
   Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,
  Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
  //动态调用 p_350.dll 中 API 函数需在此定义函数类型
  TinStr=function(x,y:integer):integer;stdcall;
var
  Form1: TForm1;
  //静态调用 p_350.dll 中 API 函数 max 需在此说明
function max(x,y:integer):integer;stdcall;external 'p_350.dll';
  //静态调用 p_350.dll 中 API 函数 jztime 需在此说明
  function jztime:string;stdcall;external 'p_350.dll';
implementation
{$R *.dfm}
  procedure TForm1.Button1Click(Sender: TObject); //静态调
//用 p_350.dll
  var       //中 API 函数 max
    x,y,z:integer;
    r:string;
  begin
    x:=1;
    y:=2;
    z:=max(x,y);    // 静态调用 API 函数 max
    showmessage(jztime+' max:'+inttostr(z));
  end;
  procedure TForm1.Button2Click(Sender: TObject); //动态调
//用 p_350.dll
  var   //中 API 函数 Min
    order:integer;
    pfunc:TFarProc;
    moudle:Thandle;
  begin
  Moudle:=Loadlibrary('p_350.dll'); //指定 p_350.dll 装入内存
    if Moudle<HINSTANCE_ERROR then exit;
 pfunc:=getprocaddress(moudle,'min'); // 得到 Min 函数地址
    order:=TInstr(pfunc)(13,32);  // 动态调用 Min 函数
    showmessage(jztime+' min:'+inttostr(order));
    Freelibrary(moudle); //从内存移出 p_350.dll
  end;
  end.
```

其运行如图 2 所示。

图 2 试例运行示意图

3 结语

实例在 Windows XP+Delphi7.0 环境下成功运行，是一种完整的在 Delphi 中制作 DLL 动态链接库及调用其中 API 函数方法技术，也提供了输出 DLL 动态链接库中公共变量值的有效方法，以供软件技术人员参考。

参考文献

[1] 东方人华，吕伟臣. Delphi7.0 入门与提高. 清华大学出版社，2003，06.

（收稿日期：2013-01-06）

（上接第 32 页）

以上代码在 VS2003 中调试通过。

3 结语

通过动态链接库将软件“忙状态”信息显示处理逻辑进行了封装，使开发人员可以将主要精力用于软件主体功能开发，提高开发效率；对用户而言，也可在软件处于“忙”状态时，从界面信息显示上知道软件在干什么，避免不必要的误解。该设计在使用后取得了良好的效果。

（收稿日期：2013-01-14）

通用、高效的 Web 上传组件的实现

汪永松

摘　要：从开发者的角度，介绍了 Web 上传组件的实现机制及改造过程，阐述了实现机制及改造思路以及通用性和高效性。

关键词：Web 上传组件；大文件上传；RFC1867 协议；HTTP 协议

1　概述

对于 Web 开发者而言，Web 上传组件是基础组件；借助上传组件，用户可以通过 Web 页将文档、数据、多媒体等文件上传到服务端；对于上传的文件，常见的应用模式有：

(1) 通过 HTTP 方式提供下载链接（例如：下载附件）。

(2) 存入到数据库（一般作为二进制块进行存放）。

(3) 对文档中的数据进行抽取（例如：电子表格文件）。

一般受到 Web 应用存在一定的要求制约（主要有：必须在 Web 页的范围内执行任务、Web 页等待时间不宜过长等），大多数 Web 系统对上传文件的大小进行了限制，即只能接收上传较小的文件。但是，在一些特定的应用中，例如：基于 Web 的多媒体资源管理，则需要上传较大的媒体文件。

通用表现一般在两个方面：第一是上传文件大小的通用性，既能处理小文件也能处理大文件；第二是上传文件数量的通用性，既能支持单文件上传也能支持多文件上传。而高效则针对的是上传文件的效率，能够在保证 Web 服务器性能稳定的基础上，提高传输效率，缩短上传时间。

1.1　Web 上传组件实例

采用多文件上传方式，上传的文件将保存在应用程序服务器指定的文件夹中，并以 HTTP 方式提供下载链接。图 1 是应用程序服务器上传文件保存文件夹中的内容。

图 1　上传文件夹中内容

图 2 是上传图 1 中 4 个音频文件（mp3）的信息输出，其中包括：各部分的分界字符串、开始上传时间、结束上传时间和上传文件信息（Web 页中变量名和文件名）。读者不难看出，4 个音频文件的总大小约为 12.7MB，耗费的时间约为 1 秒。

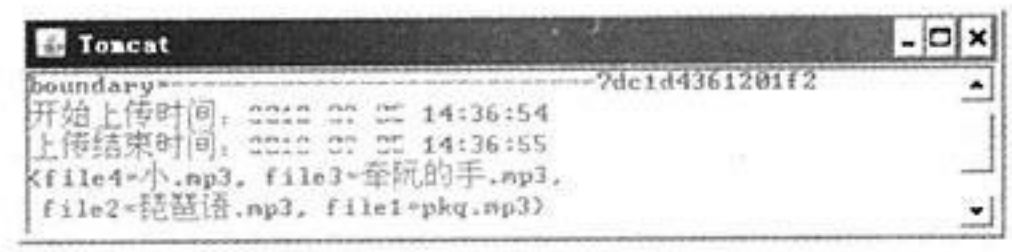


图 2　上传输出信息 1

图 3 是上传图 1 中 4 个视频文件（rmvb）的信息输出，其中包括：各部分的分界字符串、开始上传时间、结束上传时间和上传文件信息（Web 页中变量名和文件名）。读者不难看出，4 个视频文件的总大小约为 1.57GB，耗费的时间约为 185 秒（3 分 5 秒）。

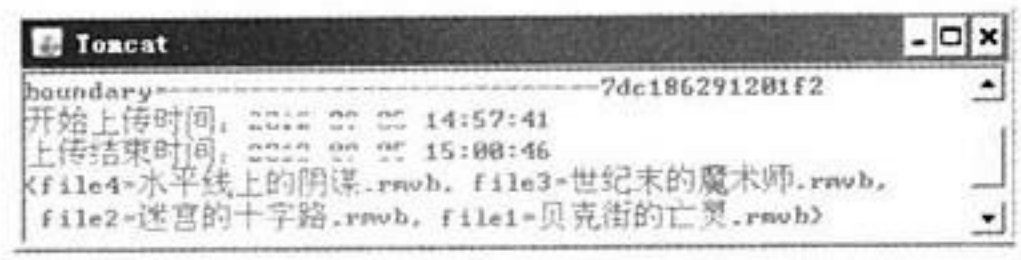


图 3　上传输出信息 2

需要说明的是，开发测试环境为：Windows XP、Tomcat 6、IE 8；双核 CPU，主频 2.0GHz、内存 3GB、硬盘 250GB。

1.2　Web 上传组件的机制

Web 上传组件的机制在 RFC1867（“HTML 中基于表单的文件上传”）中进行阐述，文件上传的发起通过 HTML 页面的表单元素（“<form>”）进行提交；而客户端文件的选择是通过类型为文件的输入元素（“<input>”）来交互。图 4 中是 Web 上传组件最简单的、文件上传流程。

在图 4 中，首先，用户在浏览器中通过表单上传文件数据；其次，Web 上传组件从 HTTP 输入流中读取编码数据（上传数据会自动进行编码）到缓冲区；再次，对编码数据进行解析，抽取其中所包含的文件内容块并输出到磁盘文件；最后，处理完毕返回上传文件信息并通过页面提示上传完毕。

需要注意的是，图 4 中的流程是最为简单的文件上传流程，忽略了很多不确定的因素，例如：缓冲区的大小如何设

定、缓冲区到磁盘文件的输出如何控制等。

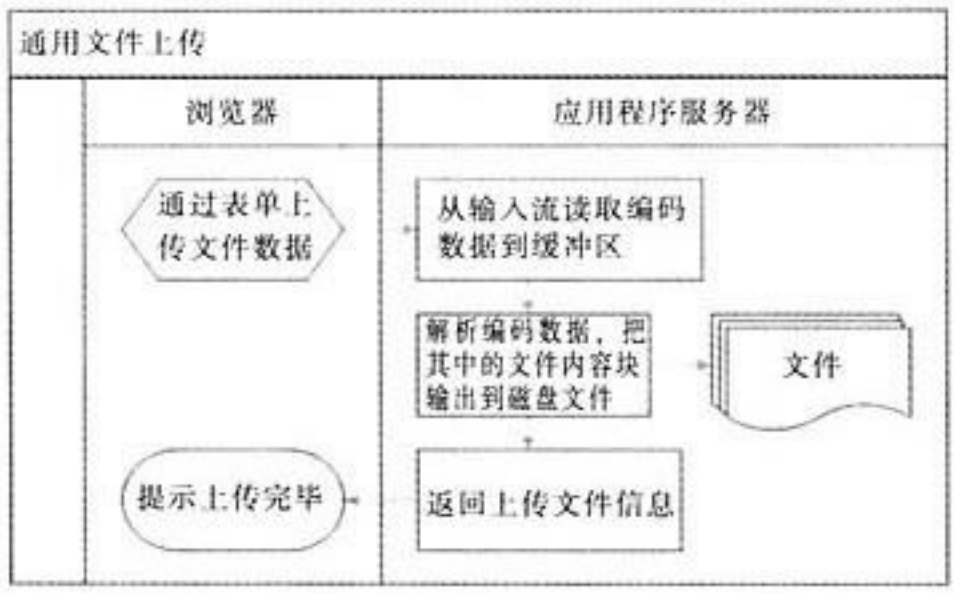


图 4 数据集成展示平台的业务架构

1.3 Web 上传组件的实现要点

根据 Web 上传组件的实现机制，读者不难看出，要实现通过 Web 上传组件上传文件，首先必须按照 RFC1867 的规范设计 HTML 表单页面；而要提高其通用性和执行效率，则可以从以下两个方面着手：

(1) 缓冲区大小设置的控制。

(2) 写入和读取缓冲区的控制。

1.3.1 上传表单页

对于上传文件的 HTML 表单页，其关键内容如代码 1 所示：

```
//代码 1  上传表单内容示例
<form  method = "POST"  ENCTYPE = "multipart/form -data"
action="<上传处理模块>">
<p>请选择文件：<input type="file" name="<变量名>" size="
30" maxlength="300"></p>
<input type="submit" value="上传">
</form>
```

代码 1 中，按照 RFC1867 标准，表单元素的方法属性必须为 “POST”，必须设置编码类型属性（“ENCTYPE”），而其动作属性（“action”）执行的模块即为 Web 上传组件；文件选取是通过类型为文件的输入元素进行交互。当点击 “提交” 按钮，浏览器会自动将文件数据进行编码上传到服务端，并由指定的 Web 上传组件进行处理。

1.3.2 缓冲区大小的设置控制

缓冲区是由应用程序服务器分配的内存空间，用于存放从客户端通过 HTTP 上传的编码数据。由于受到 Java 虚拟机及应用程序服务器的运行设置限制，应用程序服务器所分配的缓冲区大小也会受到限制。在本文的测试环境中，应用程序服务器可分配的缓冲区大小约为 65MB，当内存分配请求大余该值时，应用程序服务器将抛出 “内存不足” 的异常信息。

如此一来，缓冲区的大小设置要结合考虑应用程序服务器的制约和上传文件数据的大小。如果缓冲区的大小超过应用程序服务器的制约，会造成应用程序服务器的不稳定；如果缓冲区的大小小于应用程序服务器的制约，则会出现无法完全载入文件数据的问题；而且当上传文件内容大小远大于应用程服务器的内存制约时（例如图 3 中提到的 1.57GB），则必须考虑对文件内容的分段加载。

所以为了规避上传大文件对应用程序服务器因内存分配造成的不稳定以及降低上传模块设计的复杂度，很多 Web 上传页面都严格控制了上传文件的大小。既便如此，对于实在需要上传大文件的场合（例如：多媒体资源管理系统），则不得不使用远小于文件大小的缓冲区来对文件内容进行分段加载和处理。

1.3.3 写入和读取缓冲区的控制

从 HTTP 输入流读取上传数据时，从流传入的内容的字节数是无法确定的，所以无法采取块读取的方式，而只能采用逐个字节的方式。但是写入的控制是可以采用块的方式，而无需逐个字节写入。

使用缓冲区对超过缓冲区大小的内容进行分段加载和处理是实现 Web 上传组件对大小文件 “通吃” 以及决定上传效率的关键操作。一方面，按照 “空间换时间” 的原则，保证尽可能多的操作都在内存中进行；除了处理分配空间尽可能多的缓冲区来装载内容之外，多进行块操作也可以减少 I/O 交互，提高效率。另一方面，按照 “时间换空间” 的原则，当缓冲区大小无法满足一次性装载数据内容时，则必须考虑将已处理的数据暂存到磁盘文件；在该处理过程中，必须注意保证数据处理的完整性，因为数据分段可能会破坏一些不可分割的信息（主要包括：部分分界字符串（boundary）和各部分的头部信息（其中包括：Web 页变量名和上传文件名）），一旦这些信息被分割，则会造成全体数据处理的错乱。因此对于缓冲区的设计必须充分考虑在分段处理的模式下保证数据信息的完整性，图 5 即所设计的缓冲区结构。

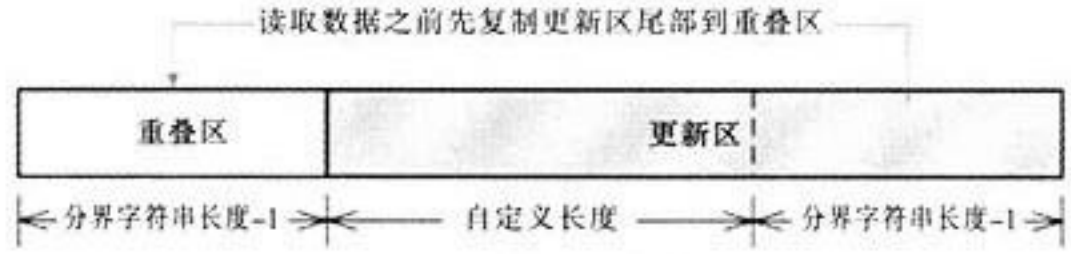


图 5 缓冲区结构

对于上传过来的数据，首先必须探测其中各部分（Part）的分界信息，即获知整体数据包含哪几个部分，每一个部分对应一个上传项。在分段载入的过程中，可能会造成一个分界信息字符串被分割到前后两次记载的缓冲区中，继而造成在缓冲区内容的处理中分界信息匹配的失败，从而导致数据解析的失败。

所以，为了避免该问题，如图 5 所示，在缓冲区的头部预留了一个重叠区用于保存缓冲区上一次内容的尾部内容，其长度为分节字符串长度减 1。每次在写缓冲区时，将掠过重叠区，而填充更新区，更新区满则将更新区内容暂存到磁盘文件，同时将更新区尾部内容复制到重叠区。如此一来，即可保

证分界字符串在缓冲区中的完整性（不在更新区范围就在重叠区+更新区范围）。

2 设计过程

2.1 主要类设计思路

2.1.1 上传组件—FooUpload3

上传组件用于读取 HTTP 输入流，并按照桩页面（Stub）传递过来的分界字符串对数据内容进行解析，继而将上传数据中的所有文件内容输出为磁盘文件，最后将文件信息返回给桩页面。其工作内容分为两部分：

（1）逐个字节读取输入流，并存入缓冲区的更新区。

（2）更新区满，则在整个缓冲区中探测分界字符串的起始位置，并将更新区内容暂存到磁盘文件。

（3）循环执行以上步骤直至输入流读取完毕。

1）若上传内容大小小于更新区容量，则直接在当前缓冲区中探测所有分节字符串的起始位置，并进入步骤（4）。

2）若上传内容大小大于更新区容量，则必须加载（2）中的临时文件，并依据（2）中所获取的分界字符串位置信息获取各部分的文件信息，继而根据文件信息提取文件数据并保存为磁盘文件，同时记录本部分的上传文件信息。

（4）根据分界信息，获取各部分的文件信息，继而根据文件信息提取文件数据并保存为磁盘文件，同时记录本部分的上传文件信息。

（5）返回各部分的上传文件信息。

2.1.2 上传文件信息—FileInfo

上传文件信息用于记录整个上传内容中各个上传文件的信息，包括：Web 变量名、文件名、文件内容的起始位置、文件内容的终止位置。不难看出，一旦获知各个上传文件在整个上传内容中的起止信息，即可完成文件内容的抽取，继而按照文件名和预定的存储路径进行文件输出。

2.1.3 上传桩页面—upload.jsp

上传桩页面，用于获取页面上下文信息（主要包括：HTTP 输入流和分界字符串），继而根据上下文信息调用上传组件，并将上传结果反馈给 Web 前端。

根据 HTTP 规范，HTTP 头部的内容类型（“content-type”）项中即包含了分界字符串信息。

2.1.4 上传表单页面—index.html

上传表单页面即按照 RFC1867 标准进行设计的 HTML 页面，用于提供与用户的操作交互。

2.2 数据库

由于文中所上传的文件仅以磁盘文件的方式进行存放，上传结果信息也只是用于提示，所以不涉及数据库的操作。但是在通常项目开发中，可以使用数据库直接存放文件内容（例如：BLOB 类型）或仅仅记录文件存储的路径信息。

3 开发过程

3.1 上传组件

3.1.1 主方法

上传组件提供了一个主要方法（“upload”）用于执行上传请求，其输入参数包括：HTTP 输入流、上传文件夹路径、临时文件夹路径和各部分的分界字符串，如代码 2 所示：

//代码 2 上传组件主方法

```
//文件名:FooUpload3.java
public Hashtable<String,String> upload(InputStream is,
      String uploadPath, String tempPath, String boundary) {
      //部分的分界信息
      final int boundaryLen = boundary.length();
      final byte[] boundaryBuf = boundary.getBytes();
 //用于存放上传文件信息(键为 Web 页中变量名,值为文件名)
Hashtable<String, String> ht = new Hashtable<String,String>();
      //缓冲区
      byte[] block = new byte[boundaryLen-1+THRESHOLD];
       try {
            //读取计数
            int count = 0;
            //分界字符串的位置信息
      ArrayList<Integer> pos = new ArrayList<Integer>();
            //读取缓冲块序号
            int blockNo = 0;
      //临时文件
      FileOutputStream fos = null;
      int abyte = -1;
      //读取输入流并转存到字节流
      while((abyte=is.read())! =-1) {
            block[boundaryLen-1+count] = (byte)abyte;
            count++;
            if(count>=THRESHOLD) { //更新区满了
                  if(blockNo==0) {
                        //第一次输出缓冲区,创建临时文件
fos = new FileOutputStream(tempPath+"temp.dat");
                  }
                  //探测分界字符串在文件中的位置
for(int i = 0; i < (boundaryLen-1+THRESHOLD); ++i) {
                  if(isStartWith(block,i,boundaryBuf)) {
                        pos.add
(blockNo*THRESHOLD+i-(boundaryLen-1));
                  }
            }
            //暂存到临时文件
fos.write(block,boundaryLen-1,THRESHOLD);
            blockNo++;
            count = 0;
            //复制更新区尾部到重叠区
            for(int i = 0; i < boundaryLen-1; ++i) {
                  block[i] = block[THRESHOLD+i];
```

```
                }
            }
        }
          //若只用到一次缓冲区(即上传内容不大于更新区,
//则无需保存到磁盘)
        if(blockNo == 0) {
            ……//上传内容大小不大于缓冲区
        } else {
            ……//上传内容大小大于缓冲区
            }
        } catch(IOException e) {
            e.printStackTrace();
        }
        return (ht);
}
```

代码 2 中，分配缓冲区大小为分界字符串长度减 1 再加一个常量，该常量值即更新区大小。上传组件逐个字节从输入流读取内容字节并存入缓冲区，待缓冲区中的更新区满，即对更新区内容进行处理（探测边界字符串的位置），然后把更新区内容暂存到临时文件，最后将更新区的尾部复制到重叠区。

需要注意的是，对于分界字符串在缓冲区的位置是一个相对位置，必须转换成绝对位置，以简化后续在临时文件流中的定位。对于分界字符串定位定位还有一种比较简单的方式：把读取内容先存入文件，再通过文件流来探测分界字符串在文件中的位置，该位置即为绝对位置。这种方式不用考虑将缓冲区分为重叠区和更新区，逻辑上简单明了，但显而易见的是，其会增加磁盘文件的 I/O 操作，效率较代码 2 中的方式低了很多。

待上传内容读取完毕，即可按照内容大小进行区别处理。

（1）上传内容不大于缓冲区

当上传内容大小不大于缓冲区时，对上传内容的解析可直接在缓冲区完成，如代码 3 所示：

```
//代码 3　上传内容大小不大于缓冲区
//文件名:FooUpload3.java
//探测分界字符串在文件中的位置
for(int i = 0; i < count; ++i) {
    if(isStartWith(block,i,boundaryBuf)) {
        pos.add(i-(boundaryLen-1));
    }
}
  //获取各部分中的文件信息(Web 页变量名+文件名+起始位
//置+大小)
for(int j = 0; j < (pos.size()-1); ++j) {
    FileInfo fileInfo = getFileInfo(block, pos.get(j), pos.get(j+
1), boundaryLen);
    if(fileInfo==null) {
        continue;
    }
    //保存文件
    saveToFile(block, uploadPath, fileInfo);
    //保存上传文件信息
    ht.put(fileInfo.getVar(), fileInfo.getFilename() );
}
```

代码 3 中，先探测各个部分边界字符串的位置，然后依据边界字符串的位置来获取各部分的文件信息，再根据各部分的文件信息提取文件内容并保存文件，并记录部分中的文件信息。

（2）上传内容大于缓冲区

当上传内容大小大于缓冲区时，必须进行分段加载，处理完毕后还必须暂存到临时文件；输出文件的内容的提取必须再次加载临时文件，如代码 4 所示：

```
//代码 4　上传内容大小大于缓冲区
//文件名:FooUpload3.java
if( (count<THRESHOLD) && (count>0) ) {
    //探测分界字符串在文件中的位置
    for(int i = 0; i < (boundaryLen-1+count); ++i) {
        if(isStartWith(block,i,boundaryBuf)) {
    pos.add(blockNo*THRESHOLD+i-(boundaryLen-1));
        }
    }
    //将最后内容写入到临时文件
    fos.write(block,boundaryLen-1,count);
}
//提交保存临时文件
fos.close();
//重新载入临时文件
        FileInputStream fis = new FileInputStream (new File
(tempPath+"temp.dat"));
//获取各部分中的文件信息(Web 页变量名+文件名+起始位
//置+大小)
for(int j = 0; j < (pos.size()-1); ++j) {
    FileInfo fileInfo = getFileInfo(fis, pos.get(j), pos.get(j+1),
boundaryLen);
    if(fileInfo==null) {
        continue;
    }
    //保存文件
    saveToFile2(fis, uploadPath, fileInfo);
    //保存上传文件信息
    ht.put(fileInfo.getVar(), fileInfo.getFilename() );
}
//关闭并删除临时文件
fis.close();
//f.delete();
```

代码 4 中，在缓冲区分段加载上传内容的过程中，即对各部分的边界字符串进行探测并转储（代码 2 中），对于最后一次读取内容也必须执行同样操作。当上传内容读取完毕，需要对临时文件的内容进行提交。

不同于代码 3，代码 4 中的获取各部分的文件信息必须先

加载临时文件，再通过文件流来获取各部分中的文件信息，输出文件的内容也需要通过文件流提取。

3.1.2 探测分界字符串的位置

代码 5 是代码 2 中探测分界字符串的起始位置的方法，其采用逐个字节匹配的方式进行整个字节数组的匹配。

代码 5　探测分接字符串的位置

文件名：FooUpload3.java

```
//判断子字节数组是否是父字节数组指定位置的开始
private boolean isStartWith(byte[] pbuf, int pos, byte[] sbuf) {
    int i = 0;
    for(; i < sbuf.length; ++i) {
        if(sbuf[i]! =pbuf[pos+i]) {
            break;
        }
    }
    return (i>=sbuf.length);
}
```

3.1.3 获取各部分的文件信息

根据代码 3 和代码 4 可知，获取各部分的文件信息存在两种情形：直接从缓冲区获取和通过文件流获取。在解析之前，读者还需要了解上传组件从 HTTP 输入流读取到的内容究竟是什么样子，图 6 中即为上传内容导出的字节块。

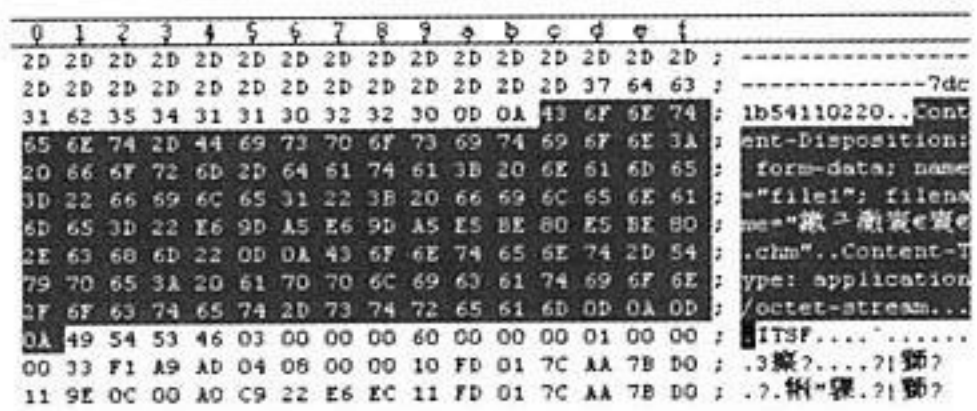


图 6　上传内容字节块

图 6 中，所选择区域即为部分的头部信息，而头部信息之前即为各部分之间的边界字符串（形如“-----7dc1b54110220”），边界字符串与头部信息之间用回车换行符（“\r\n”）进行分隔。头部之后即为文件内容块（从“ITSF”开始），文件内容块与文件头间隔了两个回车换行符，文件内容块的尾部与下一个边界字符串以回车换行符分隔。各部分的头部信息包含两行：第一行是内容描述，主要包含了 Web 也传递过来的变量名（“file1”）和文件名；第二行是内容类型。解析要做的就是从文件头部提取变量名和文件名信息。

（1）从缓冲区中获取各部分的文件信息

由于文件信息以回车换行符进行分隔，所以从各部分的头部检测前 3 个回车换行符的位置即可获取内容描述行、内容类型行和文件内容的起始位置，继而可获取内容描述行的字节内容，如代码 6 所示：

//代码 6　从缓冲区中获取各部分的文件信息

//文件名：FooUpload3.java

```
private FileInfo getFileInfo(byte[] buffer, int spos, int epos, int
boundaryLen) {
    int i = (spos+boundaryLen);
    int j = (epos-SEPARATOR.length);
    //探测 4 个位置：内容描述头部、内容类型头部、文件块头
//部和尾部
    int[] pos = new int[4];
    int k = 0;
    //探测前 3 个位置
    for(; i < (epos-SEPARATOR.length); ++i) {
        if(isStartWith(buffer,i,SEPARATOR)) {
            pos[k++]=i;
            if(k>=(pos.length-1)) {
                break;
            }
        }
    }
    //探测第 4 个位置
    for(; j>= pos[k]; --j) {
        if(isStartWith(buffer,j,SEPARATOR)) {
            pos[k++] = j;
            break;
        }
    }
    //获取内容描述行并解析
            String contentDesc = getStr(buffer,pos[0]+
SEPARATOR.length,pos[1]);
    FileInfo fileInfo = parseContentDesc(contentDesc);
    if(fileInfo==null) {
        return (null);
    }
    //文件块头部有一空行
    fileInfo.setStartPos(pos[2]+(SEPARATOR.length*2));
    fileInfo.setEndPos(pos[3]);
    return (fileInfo);
}
```

代码 6 中，对于文件内容终止位置的探测是从各部分的尾部逆向探测的，如此操作不仅避免文件内容中与回车换行符的重复，而且极大提高检索效率。最后解析并记录的文件信息包括：Web 变量名、文件名、文件内容块的起始位置和终止位置。

（2）从临时文件中获取各部分的文件信息

相比代码 6 中缓冲区的操作，从临时文件中获取各部分的文件信息需要通过文件流的读取游标进行跳转，如代码 7 所示。

//代码 7　从临时文件中获取部分中的文件信息

//文件名：FooUpload3.java

```
private FileInfo getFileInfo (FileInputStream fis, int spos, int
epos, int boundaryLen) throws IOException {
    //如果无上传项目
    if(epos <= HEADER_SIZE) {
        return (null);
    }
```

```
    //缓冲区
    byte[] buffer = new byte[HEADER_SIZE];
     //探测 4 个位置:内容描述头部、内容类型头部、文件块头
//部和尾部
    int[] pos = new int[4];
    //探测分段最后的换行块位置
    fis.getChannel().position(epos-HEADER_SIZE);
    fis.read(buffer);
    int i = (epos-SEPARATOR.length);
    for(; i>= (epos-HEADER_SIZE); --i) {
        if  (isStartWith  (buffer,i -(epos -HEADER_SIZE),
SEPARATOR)) {
            pos[3] = i;
            break;
        }
    }
    //探测前 3 行的换行块位置
    fis.getChannel().position(spos);
    fis.read(buffer);
    int j = (spos+boundaryLen);
    int k = 0;
    for(; j < (spos+boundaryLen+HEADER_SIZE); ++j) {
        if(isStartWith(buffer,j-spos,SEPARATOR)) {
            pos[k++]=j;
            if(k>=3) {
                break;
            }
        }
    }
    //获取内容描述行并解析
    String contentDesc = getStr(buffer,pos[0]-spos,pos[1]-
spos);
    FileInfo fileInfo = parseContentDesc(contentDesc);
    if(fileInfo==null) {
        return (null);
    }
    //文件块头部有一空行
    fileInfo.setStartPos(pos[2]+(SEPARATOR.length*2));
    fileInfo.setEndPos(pos[3]);
    return (fileInfo);
}
```

代码 7 中，使用了一个缓冲区用于对临时文件的读取和解析操作，其解析的顺序与代码 6 相反：先探测文件内容的终止位置，再探测 3 个头部信息位置。

3.1.4 解析文件信息

代码 8 是代码 6 中获取内容描述行以及解析文件信息的主要代码。

```
//代码 8  解析各部分所包含的文件信息
//文件名:FooUpload3.java
//从字节数组中提取字符串
private String getStr(byte[] buffer, int spos, int epos) {
    //文件头信息
                        ByteArrayOutputStream baos = new
ByteArrayOutputStream();
    String text = null;
    try {
        baos.write(buffer, spos, epos-spos);
        text = baos.toString(CHARSET);
        baos.close();
        baos = null;
    } catch(Exception e) {
    }
    return (text);
}
//解析文件描述行
//形如:Content-Disposition: form-data; name="xxx"; filena
//me="xxx"
private FileInfo parseContentDesc(String contentDesc) {
    //变量名
    int pos1 = contentDesc.indexOf(TAG1);
int pos2 = contentDesc.indexOf('\"',pos1+TAG1.length());
    //文件名
    int pos3 = contentDesc.indexOf(TAG2);
int pos4 = contentDesc.indexOf('\"',pos3+TAG2.length());
    //
    final String var = contentDesc.substring (pos1 +TAG1.
length(), pos2);
    final String filename = contentDesc.substring (pos3 +
TAG2.length(), pos4);
    //若文件名为空则掠过
    if(filename.length()<1) {
        return (null);
    }
    FileInfo fileInfo = new FileInfo(var, filename);
    return (fileInfo);
}
```

代码 8 中需要注意的是，文件名的编码可能是中文（如图6 中所示），需要采用 GBK 编码，否则获取到的文件名将是乱码。

3.1.5 文件内容的输出

各部分中文件内容的存储形式决定了其输出的不同，输出方式也将分为：从缓冲区输出和通过文件流输出。

（1）从缓冲区输出文件内容

从缓冲区输出文件内容可以“一步到位”，及批量将文件内容块输出到文件输出流，如代码 9 所示：

```
//代码 9  从缓冲区输出文件内容
//文件名:FooUpload3.java
private void saveToFile (byte [] buffer, String path, FileInfo
fileInfo) throws IOException {
    //初始化文件夹
```

```
        File f = new File(path);
        if(! f.exists()) {
              f.mkdir();
        }
        //写入到文件
        FileOutputStream fos = new FileOutputStream (path+
fileInfo.getFilename());
        fos.write(buffer, fileInfo.getStartPos(), fileInfo.getLen());
        fos.close();
}
```

代码9中，根据文件信息中的文件内容块的起始位置和长度信息，一次性地将文件内容块输出到文件。

(2) 从临时文件流输出文件内容

和临时文件的分段写入同理，从临时文件输出文件内容也需要分段写入，而所谓的分段即需要借助缓冲区的帮助，如代码10所示：

```
//代码 10  从临时文件流输出文件内容
//文件名:FooUpload3.java
private void saveToFile2 (FileInputStream fis, String path,
FileInfo fileInfo) throws IOException {
        //初始化文件夹
        File f = new File(path);
        if(! f.exists()) {
              f.mkdir();
        }
        //写入到文件
        FileOutputStream fos = new FileOutputStream (path+
fileInfo.getFilename());
 final int blockCount = (int) (fileInfo.getLen()/THRESHOLD);
        fis.getChannel().position(fileInfo.getStartPos());
        byte[] buffer = new byte[THRESHOLD];
        for(int i = 0; i < blockCount; ++i) {
              fis.read(buffer);
              fos.write(buffer,0,THRESHOLD);
        }
        //缓冲区中剩余字节也需要写入
        final int bytes = fis.read(buffer);
        if(bytes ! = -1) {
              fos.write(buffer,0,bytes);
        }
        fos.close();
}
```

代码10中，写入到文件输出流之前先根据文件内容块的大小计算其需要分几次写入，然后再分段地读取文件内容，再写入到输出文件。

3.2 上传文件信息

通过以上代码，读者应该可以了解上传文件信息包含哪些信息：Web变量名、文件名、文件内容块的起始位置和终止位置，如代码11所示：

```
//代码 11  上传文件信息
//文件名:FileInfo.java
public class FileInfo {
        private String var;
        private String filename;
        private int spos;
        private int epos;
        ……
        public int getLen() {
              return (this.epos-this.spos);
        }
};
```

代码11中，文件内容块的大小是通过终止位置减起始位置而得到。

3.3 文件上传桩页面

上传桩页面是一个JSP页面，其可以从HTTP请求信息中获取上传内容的分界字符串信息，继而通过分接字符串调用上传组件，获取上传结果，其主要过程如代码12所示：

```
//代码 12  文件上传桩页面
//文件名:upload.jsp
<%
        //需要设置 JSP 容器的 URIEncoding="utf-8"
        request.setCharacterEncoding("utf-8");
        InputStream is = request.getInputStream();
        //获取 boundary
        //multipart/form-data; boundary=xxx
      final String val = request.getHeader("content-type");
        int spos = val.indexOf(BOUNDARY);
        if(spos == -1) {
              return;
        }
        spos += (BOUNDARY.length());
        final String boundary = val.substring(spos);
        final Hashtable <String,String > ht = FooUpload3.
getInstance().upload(is, uploadPath, tempPath, boundary);
        …
%>
```

代码12中，上传内容的边界字符串信息在HTTP请求的头部的“内容类型”项中解析获取。

3.4 上传表单页面

上传表单页面即按照RFC1867的规范所定义的HTML网页，如代码13所示。

```
//代码 13  上传表单页面
//文件名:index.html
<form  method = "POST"  ENCTYPE = "multipart/form -data"
action="upload.jsp">
<p>请选择文件:<input type="file" name="file1" size="30"
maxlength="300"><br>
```

(下转第51页)

基于 Excel VBA 的会计单据演示训练系统开发（一）

何 燕

摘 要：在财经类专业的会计学课程教学中，为使学生熟练掌握常用会计单据的填写方法，需要进行模拟训练，而传统的手工模拟训练方式存在着一系列缺陷，难以适应会计教学及学习信息化发展方向的需要。为满足这一需要，运用 Excel2007 软件在 Excel VBA 平台下开发完成了会计单据演示训练系统。

关键词：Excel VBA 编程；原始凭证；模拟训练；系统开发

为使财经类专业学生掌握常用会计单据（或称为原始凭证）的正规填写方法，会计教学中传统的做法是进行手工模拟训练。这一方法的缺点：（1）需购买或印制大量的凭证模拟材料，开支较大，管理不便；（2）模拟教学中，教师需讲解、演示、纠错辅导并管理，工作量较大；（3）纸质模拟材料填写错误后不得不换新的，往往浪费较多；（4）模拟训练课程时间有限，难以完成所有常用会计单据的填写训练；（5）模拟训练课程难以满足学生随时进行填写训练的需要；（6）纸质模拟材料携带不便，保存时间有限。综上所述，会计教学中需要一种新的会计单据训练模式，即在计算机条件下开发会计单据训练演示系统以解决传统训练模式下的问题。

1 设计思路

（1）新建一个 Excel 工作簿，将其命名为“原始凭证训练演示系统”（后简称“系统”），建立若干个工作表，表名分别为“主界面”、“训练题”、“支票”、“增值税发票”及其他原始凭证工作表。

（2）在“主界面”工作表中设置各原始凭证训练选项，在“训练题”工作表中存放各原始凭证训练题库，“支票”工作表是为实现支票的训练演示、“增值税发票”工作表是实现增值税发票的训练演示，其他工作表实现其他单据训练。

2 制作方法

2.1 “主界面”工作表的制作

（1）选中 A1：P7 单元区域将其合并单元格，并设置背景色。

（2）将 A8：P44 单元区域合并并设置背景色。

（3）插入艺术字“原始凭证训练演示系统”，编辑调整艺术字大小及位置使其大体覆盖 D1：M7 区域。

（4）插入图案覆盖 A1：C7 区域。

（5）在 A9：C10 区域插入横排文本框“支票填写训练”，如图 1 所示，在别的位置插入其他对应文本框并编辑。

图 1 主界面工作表

2.2 “训练题”工作表制作

如图 2、图 3 所示。

（1）每一训练题库的起始单元格是 A1、AA1、BA1、CA1、DA1……，如支票训练题库起始于 A1；增值税发票训练题库起始于 AA1；后续其他训练题库起始于 BA1、CA1、DA1。

（2）每一训练题库的名称在第一行，如“支票训练题库”字样在第一行。第二行是信恒保温瓶厂的基本简介，简介中包含相应原始凭证训练所需的基本资料。第三行是样题，第四行及其后为训练题。

（3）每一训练题库的第一列内从第四行往下依次输入题号 1、2、3、4……。第二列皆为合并区域，样题位于“B3”或“AB3”、“BB3”、“CB3”、“DB3”……。（该题库根据立信出版社出版由张维宾、姚津编写的《新编会计模拟实习——工业企业分册》第四版中的相关模拟实训题而建立）。

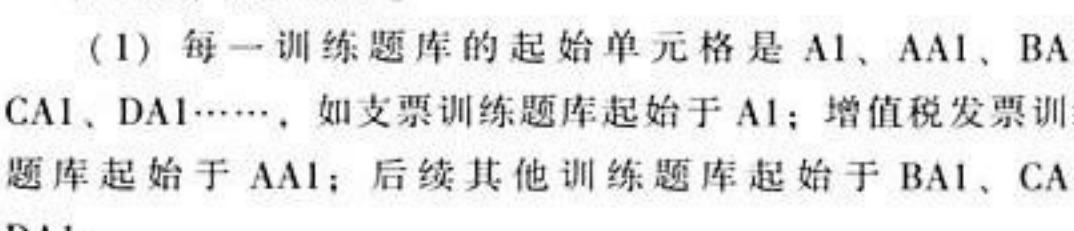


图 2 训练题工作表——支票训练题库

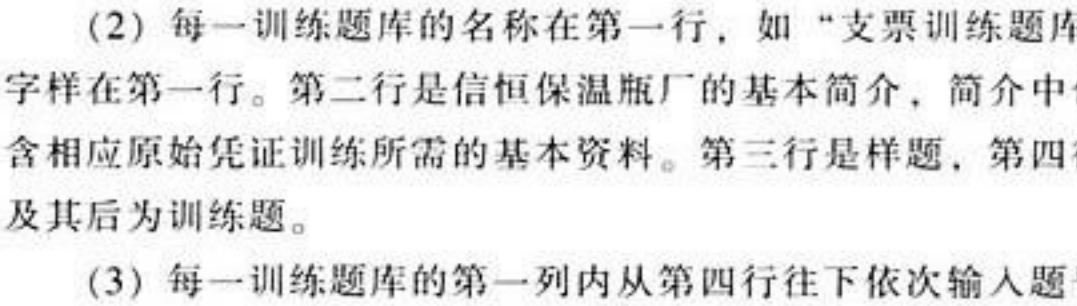


图 3 训练题工作表——增值税发票训练题库

2.3 “支票”工作表制作

如图4、图5所示。

（1）将A2：AD2区域合并，并调整至适当行高，在A2内输入“=训练题! A2”。

（2）在B1内输入“=COUNTA (训练题! A4:A65536)”以计算出除了样题以外支票训练题数目，将B1内字体颜色设为“白色”以遮盖住公式。

（3）在M3内输入“=" 除样题外,本练习共有" &COUNTA (训练题! A4:A65536) &" 条业务题可供选择！你可根据业务题或自由填写支票!"”，将M3内字体定为“红色”，以提示题库中的支票训练题数目。

（4）将B4：AD5区域合并为单元格B4，在B4内输入“=IF (A4="" ,"" ,IF (A4=0,训练题! B3,INDEX (训练题! B4:B65536,A4)))”，以显示相应的训练题或样题文字。

（5）插入一窗体控件，在F3：I3区域画出该控件，在随后出现的指定宏窗体中点击“确定”，暂不编辑宏程序。在控件中编辑文字，输入“业务题选择”，选择该文字，编辑字体。同样方法在A7：M8区域插入9个控件并进行编辑，如图4所示。

（6）在A9：AE27单元区域制作支票票面如图4所示。将AA11（AA11：AD11合并）、C11（C11：H11合并）单元格格式设为文本格式。

（7）如图5中，B28：B38内输入“¥、0、1、2……9”。W28（合并了W28：Z28区域）至W47内是2001至2020年的阿拉伯数字字样，AA28（合并了AA28：AC28区域）内输入公式：

图4 支票工作表——支票填写练习部分

图5 支票工作表——票面练习部分的辅助数据

（注：39~46、48~57行因篇幅缘故隐藏，并且该表制作完成后应去除网络线并将字体颜色改为白色以遮盖该区域并美化界面）

“=TEXT (MID (W28,1,1) ," [DBNum2]") &TEXT (MID (W28,2,1) ," [DBNum2]") &TEXT (MID (W28,3,1) ," [DBNum2]") &TEXT (MID (W28,4,1) ," [DBNum2]")”，即将“2001”翻译为中文大写“贰零零壹”，用AA28单元格的下拉填充柄拉至AA47以完成所有年份的翻译。

（8）在H28至H58区域内输入每月1-31日的数字字样，在G28内输入“=IF (LEN (H28) =1," 零" &TEXT (H28," [DBNum2]") ,TEXT (H28," [DBNum2]"))”，将“1”翻译成“零壹”，用填充柄拉至G58完成每月1-31日的大写翻译。

（9）在L33内输入：“=TEXT (MID (C12,1,1) ," [DBNum2]") &TEXT (MID (C12,2,1) ," [DBNum2]") &TEXT (MID (C12,3,1) ," [DBNum2]") &TEXT (MID (C12,4,1) ," [DBNum2]")”，将支票票面上存根部分的年份数字翻译成中文大写。

（10）在M33内输入“=IF (LEN (E12) =1," 零" &TEXT (E12," [DBNum2]") ,TEXT (E12," [DBNum2]"))”，将支票存根部分的月份数字翻译成中文大写。在M34内输入“=IF (LEN (G12) =1," 零" &TEXT (G12," [DBNum2]") ,TEXT (G12," [DBNum2]"))”，将存根部分的日数字翻译成中文大写。

（11）打开VASUAL BASIC编辑器。双击“工程资源管理器”窗口中的“模块”——“模块1”，打开其代码窗口。点击“插入”菜单中的“过程”命令，选择“函数”类型、“公共的”范围，输入名称“金额数值”。并在代码框中进行函数过程编辑如下：

```
' 本函数将正规单据中细格型金额转为数值型金额
Public Function 金额数值(myrange As Range) As Double
Dim c As Integer, r As Integer, colbeggin As Integer, colend
_ As Integer, sum As Double, k As Integer
c = myrange.Columns.Count
colbeggin = myrange.Cells(1).Column
colend = myrange.Cells(myrange.Count).Column
r = myrange.Row
k = -2
For i = colend To colbeggin Step -1
 h = Cells(r, i).Value
If Cells(r, i).Value = "¥" Or Cells(r, i).Value _
= "" Then
      h = 0
End If
sum = sum + h * 10 ^ (k)
k = k + 1
Next i
金额数值 = sum
End Function
```

同法，插入并编辑函数“金额中文大写”[1]：

```
' 本函数将金额数值转为大写中文字样
Public Function 金额中文大写(金额数值 As _
Double) As String
Dim a As Variant, b As Integer, c As Integer
Dim q(1 To 9) As String, s As String
q(1) = "壹": q(2) = "贰": q(3) = "叁": q(4) = "肆"
q(5) = "伍": q(6) = "陆": q(7) = "柒": q(8) = "捌": q(9) = "玖"
a = Int(金额数值)
```

```
b = Val(Mid(Str(金额数值), InStr(1, Str(金额数值), ".") + 1, 1))
c=Val(Right(Application.WorksheetFunction. _
Text(Str(金额数值 * 100), "0"), 1))
 s=Application.WorksheetFunction.Text( a, "[DBNum2]")
 If a = o Then
    If b = 0 Then
       If c = 0 Then
          金额中文大写 = ""
          Exit Function
       Else
          金额中文大写 = q(c) & "分"
          Exit Function
       End If
    ElseIf c = 0 Then
          金额中文大写 = q(b) & "角整"
          Exit Function
    Else
       金额中文大写 = q(b) & "角" & q(c) & "分"
       Exit Function
    End If
ElseIf b = 0 Then
    If c = 0 Then
          金额中文大写 = s & "元整"
          Exit Function
    Else
          金额中文大写 = s & "元零" & q(c) & "分"
          Exit Function
    End If
 ElseIf c = 0 Then
    金额中文大写 = s & "元" & q(b) & "角整"
    Exit Function
 Else
    金额中文大写 = s & "元" & q(b) & "角" & q(c) & "分"
Exit Function
 End If
End Function
```

（12）在图 5 的 C28 格（由 C28：E28 合并，并且 C28 单元格式采用两位小数的数字格式）输入“=金额数值 (T15:AC15)”（T15：AC15 为支票票面的金额细格填写区域，该区域中的每一格为合并单元格，如 T15 格是 T15：T16 区域合并），这样可将支票细格金额区填写的数字转换为标准的金额数值。在 L28（由 L28：T31 合并）内输入“=金额中文大写(C28)”，这样将金额数值翻译为标准的会计金额中文大写字样。

（13）选中 M12 单元格（为 M12：O12 区域的合并单元格），运用“数据有效性”命令，设定 M12 格只能输入如 AA28：AA47 区域的“贰零零壹”至“贰零贰零”间的年份中文大写，并在选定 M12 格时会出现“中文大写数字写法：壹、贰、叁、肆、伍、陆、柒、捌、玖、零”的提示内容，且在输入无效数据时就会出现“年份应在贰零零壹至贰零贰零范围内，并且要符合中文数字大写规范！”的错误提示。

（14）运用“数据有效性”命令，设置 Q12 格（Q12 与 R12 格合并）只能输入如 G28：G39 区域内的“零壹”至“壹拾贰”间的月份中文大写字样，选定 Q12 格时显示提示框提示“书写提示如：零壹、零贰、零叁、零肆、零伍、零陆、零柒、零捌、零玖、壹拾、壹拾壹、壹拾贰”，在输入无效数据时出现错误提示框显示“书写不规范或超出正常范围！”。

（15）运用“数据有效性”命令，设置 T12 格（T12 与 U12 合并）只能输入如 G28：G58 区域内“零壹”至“叁拾壹”间的日期中文大写，选定时显示“输入规范如下：零壹、零贰、零叁、.....叁拾、叁拾壹”，输入无效数据时出现错误提示“书写不规范或已超出正常范围！”。

（16）运用“数据有效性”命令，设置 C12 格只允许输入 2001~2020 间的整数，在输入无效数据时出现错误提示框“应输入年份阿拉伯数字，且年份在 2001~2020 范围！”。

（17）运用“数据有效性”命令，设置 E12 格只允许输入 1~12 间的整数，在输入无效数据时出现错误提示框“应输入月份阿拉伯数字，且月份在 1~12 范围！”。

（18）G12 格只允许输入 1~31 间的整数，设置错误提示框显示“应输入日期阿拉伯数字，且日期在 1~31 范围！”。

（19）M14 格（M14：S16 合并）设置输入提示框“大写金额书写规范如：¥3009——叁仟零玖元整；¥780.09——柒佰捌拾元零玖分；¥19.90：壹拾玖元玖角整；¥100000.90：壹拾万元玖角整”。

（20）T15：AC15 区域（细格金额填写区）中的每一细格只允许输入如 B28：B38 区域的的金额符号或一位整数，设置输入提示框“在金额前要加“¥”！输入规范如：¥10080.90、¥45800.00”，设置错误提示框“金额明细格中的每格只能输入一位整数！”（选定 T15：AC15 区域，进行“数据有效性”设置，最后在“设置”标签内同时按 SHIFT+CTRL+ENTER 完成整个区域的有效性设置）。

（21）将支票填写时涉及的单元格区域取名为“支票未锁定区域”，如图 4，这些区域包括 C11, C12, E12, G12, C21,C22, C23, M12, Q12, T12, M13, AA11, AA12, AA13, M14, T15, U15, V15, W15, X15, Y15, Z15, AA15, AB15,AC15,M18,F26, F27。

（22）当支票票面填写完成进行保存时检验，要求某些区域内文字内容不得为空值，所以将该区域命名为“zp 文字不空区域”。这些区域包括 C11, C12,E12,G12,C21,C22,C23,F26, F27, M12, Q12, T12, M13, AA11, AA12, AA13, M14, M18。

参考文献

[1] 韩小良，韩舒婷. Excel VBA 从入门到精通. 中国铁道出版社.

（收稿日期：2013-01-08）

在 ASP.NET 中利用 DetailsView 控件实现数据的综合处理

畅育超

摘　要：通过在 ASP.NET2.0 中，利用 DetailsView 控件实现数据的查询、编辑、更新、删除等操作。

关键词：DetailsView 控件；模板列；VB.NET+SQL Server 开发；ADO.NET 编程

1　引言

DetailsView 控件的主要功能是以表格形式显示和处理来自数据源的单条数据记录，其表格只包含两个数据列。一个数据列显示数据列名，另一个数据列显示与其对应列名相关的详细数据值。这种显示方式对于数据列较多，需要逐行显示详细数据的情况非常有用。DetailsView 控件通常可与 GridView 控件结合使用，以实现主细表信息的呈现。

DetailsView 控件具有分页、更新、插入或者删除记录的功能，相对于其他的列控件来说，模板列可以处理更加复杂的数据呈现，可以在一个 DetailsView 列中显示多个数据列，但 DetailsView 控件的输入仍然是把每个字段都以 HTML 标记<table>的形式显示成一行，显得比较别扭。因此，该控件常用于数据的呈现、修改、更新、删除等操作。在此以具体实例来讲解 DetailsView 控件的相关功能的实现。

2　基础知识与技巧

2.1　自定义字段

在 DetailsView 控件中，AutoGenerateRows 属性的默认值设置为 true，它为数据源中某个可绑定类型的每个字段自动生成了一个绑定字段的对象，按其 SQL 语句的字段顺序，每个字段以文本形式显示在一行中。自动生成行提供了一种显示记录中每个字段的快速简单的方式。

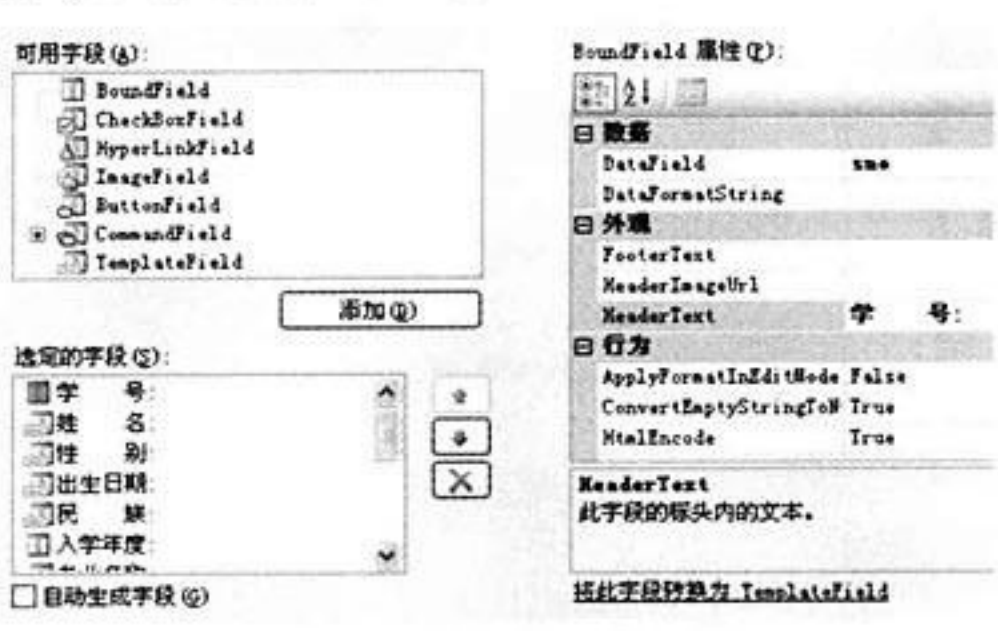


图 1　DetailsView 控件设置

而要使用其高级功能，就必须显式声明要包含在 DetailsView 控件中的行字段。若要声明行字段，首先必须将 AutoGenerateRows 属性的值设置为 False，然后在 DetailsView 控件开始和结束标记之间添加<Fields>标记。最后列出想包含在<Fields>之间的行字段，或者通过编辑字段界面手动添加字段，如图 1 所示。

在图 1 中选择添加“BoundField”后，修改其主要属性“HeaderText”和“DataField”即可。

2.2　空格和字段截取的方法

2.2.1 空格处理

例如在数据库中定义“地址”字段的长度为 50，输入时可能只有 20 位，而保存时数据库就自动在其后利用空格补齐 50 位，但在编辑时就在控件内显示包括空格在内的 50 各字符，这时就不能再追加数据了，所以要求数据显示在控件中以前就把多余的空格取掉，方法如下：

先把相应字段转换为模板列，然后在相应字段的编辑模板（EditItemTemplate）中找到对应的控件（如 TextBox），在该控件的 DataBindings 属性中选择“自定义绑定”，在此输入：

eval (" address ")) .tostring.trim () 即可。

2.2.2 字符串截取处理

为了使版面布局合理，特别是字段对应的值显示的长度基本一致，在不影响内容可读性的情况下，要求字段的值在显示特定的长度，其余部分截取，方法如下：

先把相应字段转换为模板列，然后在相应字段模板（ItemTemplate）中找到对应的控件（如 Lable），在该控件的 DataBindings 属性中选择“自定义绑定”，在此输入：(eval (" address")) .tostring () .substring (0,15) 即只显示 15 个字符。

2.3　DropDownList 控件赋值问题的解决思路

（1）一般情况下，为了特殊数据（性别、民族、状态、学历等）输入的准确性、一致性，通常利用 DropDownList 控件作为输入数据的控件，而且这类特殊数据作为 DropDownList 控件的数据源一般情况是不变的，即为静态，数据源一般都是在前置页面直接加载而不需要访问数据库。关于这类特殊数据在编辑状态下如何正确呈现数据本身的信息，即“性别”字段在数

据库中的值是“女”，但编辑状态下 DropDownList 控件页面前置加载的数据源是“男”和“女”，如何让 DropDownList 控件正确呈现当前数据库中值“女”是问题的关键，解决思路：先在 DetailsView 控件模板列的 EditItemTemplate（例如“性别”字段）模板添加一个 DropDownList 控件和一个隐含字段 HiddenField，隐含字段 HiddenField 用来保存当前记录中性别的值，而 DropDownList 控件保存的是性别的所有值，然后通过循环在 DropDownList 控件数据源中找到隐含字段中保存的值，然后设置 DropDownList 控件的 Selected 属性为 True 即可。具体代码如下：

```
    '给模板列中的 DropDownList 控件赋值(即把数据库中
'的值赋给静态的 DropDownList)
    ap = Details_1.Rows(6).Cells(1).FindControl("hfd_zy")
'取得专业隐藏字段的值
    bjf_1 = Details_1.Rows(7).Cells(1).FindControl("hfd_bj")
'取得班级隐藏字段的值
    xlf_1 = Details_1.Rows(11).Cells(1).FindControl("hfd_xl")
'取得学历隐藏字段的值
    ztf_1 = Details_1.Rows(12).Cells(1).FindControl("hfd_zt")
'取得状态隐藏字段的值
    xbf_1 = Details_1.Rows(2).Cells(1).FindControl("hfd_xb")
'取得性别隐藏字段的值
    mzf_1 = Details_1.Rows (4).Cells (1).FindControl ("
hfd_mz") '取得民族隐藏字段的值
    xl_down = Details_1.Rows (11).Cells (1).FindControl("
DrList2") ' 获得 DropDownList 控件标示符(学历)
    zt_down = Details_1.Rows (12).Cells (1).FindControl("
DropDownList2") ' 获得 DropDownList 控件标示符(状态)
    xb_down = Details_1.Rows (2).Cells (1).FindControl("
DropDownList1") ' 获得性别值
    mz_down = Details_1.Rows (4).Cells (1).FindControl("
DropDList2") ' 获得民族值
    Dim i, j, p, q As Integer ' 循环变量
    For i = 0 To xl_down.Items.Count - 1
      If (xl_down.Items (i).Value.Trim = xlf_1.Value.Trim)
Then
        xl_down.Items(i).Selected = True' 找到后选定
        Exit For
      Else
        xl_down.Items(i).Selected = False
      End If
    Next
    For j = 0 To zt_down.Items.Count - 1
      If (zt_down.Items (j).Value.Trim = ztf_1.Value.Trim)
Then
        zt_down.Items(j).Selected = True' 找到后选定
        Exit For
      Else
        zt_down.Items(j).Selected = False
      End If
    Next
    For p = 0 To xb_down.Items.Count - 1
      If (xb_down.Items(p).Value.Trim = xbf_1.Value.Trim)
Then
        xb_down.Items(p).Selected = True' 找到后选定
        Exit For
      Else
        xb_down.Items(p).Selected = False
      End If
    Next
    For q = 0 To mz_down.Items.Count - 1
      If (mz_down.Items (q).Value.Trim = mzf_1.Value.
Trim) Then
        mz_down.Items(q).Selected = True' 找到后选定
        Exit For
      Else
        mz_down.Items(q).Selected = False
      End If
    Next
  End Sub
```

（2）当 DropDownList 控件数据源来自数据库动态加载时，DropDownList 控件如何正确呈现当前记录字段值的解决思路：先在 DetailsView 控件模板列的 EditItemTemplate（例如“专业”字段）模板添加一个 DropDownList 控件和一个隐含字段 HiddenField，隐含字段 HiddenField 用来保存当前记录中专业字段的值，而 DropDownList 控件保存的是来自数据库动态加载专业字段的所有值。然后使当前隐含字段的值与 DropDownList 控件 SelectedValue 属性所显示的值相等即可。具体代码如下：

```
Public Sub Drop_getdata (ByVal sp As DropDownList, ByVal
st As DropDownList) ' 为 Detailsview 控件列模板中的DropDo
'wnList 控件赋值函数(即动态赋值,DropDownList 控件数据源
' 来自数据库动态加载)
    Dim sqlcon As New SqlConnection()
      sqlcon.ConnectionString = ConfigurationManager.
ConnectionStrings("sqlConnectionString").ConnectionString
    sqlcon.Open()
    Dim strsql As String
    Dim strsql_1 As String
    strsql = " select distinct specialityname from tablspec
order by specialityname "
    strsql_1 = " select distinct classname from tablclass
order by classname "
    Dim cmd As New SqlCommand(strsql, sqlcon)
  Dim kcreader As SqlDataReader = cmd.ExecuteReader()
    sp.DataTextField = "specialityname" 'Detailsview 控件
' 列模板中的 DropDownList3 控件所显示表的字段(专业名)
    sp.DataSource = kcreader ' 设置 DropDownList3 控件
' 的数据源为 kcreader
    sp.DataBind() ' 绑定数据源
      sp.SelectedValue = ap.Value ' 把数据源"专业"的值赋
```

```
'给模板列 DropDownList3 控件
        kcreader.Close()
        Dim cmd_1 As New SqlCommand(strsql_1, sqlcon)
                Dim kcreader_1 As SqlDataReader = cmd_1.
ExecuteReader()
        st.DataTextField = "classname" 'Detailsview 控件列模
'板中的 DropDownList4 控件所显示表的字段(班级名)
         st.DataSource = kcreader_1 '设置 DropDownList4 控
'件的数据源为 kcreader
        st.DataBind() '绑定数据源
           st.SelectedValue = bjf_1.Value '把数据源"班级"的值
'赋给模板列 DropDownList4 控件
        kcreader_1.Close()
        sqlcon.Close()
          End Sub
```

2.4 在模板列中日期控件的使用方法

在本实例中学生的“出生日期”字段可能要进行编辑，但是 Visual.Studio.net.2005 中提供的 Calendar 日期控件在页面布局时不方便，所以采用引入外部控件来解决日期控件的布局问题，首先在工程文件中“添加引用”，选择本实例中的文件夹“rq”，然后在前置页面添加如下代码即可。

```
<asp:TemplateFidld HeaderText="出生日期:">
    <EditItemTemplate>
        <asp:TextBox ID="TextBox_brday" runat="server"
        onfocus="WdatePicker({dateFmt:'yyyy-M-d'})"
        Width="160px" Text='<%# Bind("brithday","{0:d}")%
        >'></asp:TextBox>
    </EditItemTemplate>
</body>
<script language="javascript" type="text/javascript" src="rp/
WdatePicker.js"></script>
</ html>
```

当 DetailsView 控件处于编辑模式时，单击“出生日期”修改时就会出现如图 2 所示的效果，而且不会改变当前页面的布局。

图 2 日期控件

3 功能实现

3.1 数据查询

（1）该部分是以查询的结果作为 DetailsView 控件的数据源，而 DetailsView 控件的数据源即数据绑定有两种形式即使用 DataSourceID 或使用 DataSource。使用 DataSourceID 可以将 DetailsView 控件绑定到数据源控件，例如常用的 SqlDataSource 控件、AccessDataSource 控件等。当使用 DataSourceID 属性绑定到数据源时，DetailsView 控件支持双向数据绑定。因此除了可以使该控件显示数据之外，只需要设置相关的属性，还可以使他自动支持对绑定数据的分页、插入、更新和删除操作；使用 DataSource 允许绑定到各种对象，包括 ADO.NET 数据集、数据读取器以及内存中的结构（集合）。采用此方法，需要为所有功能（修改、更新、分页、删除）编写相应代码。此实例采用 DataSource 作为 DetailsView 控件的数据源。查询主界面如图3 所示。

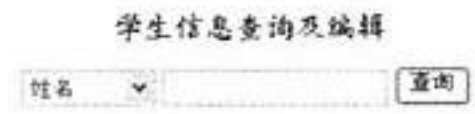


图 3 查询界面

（2）此查询可按照学生的姓名、学号、入学年度、所在班级等字段进行相应查询，主要后台代码如下：

```
Protected Sub Button_cx_Click (ByVal sender As Object,
ByVal e As System.EventArgs) Handles Button_cx.Click
'查询按钮事件
        r = Text_tj.Text.ToString.Trim
        s = DDList_1.Text.ToString.Trim
        serch_data(s, r) '调用查询模块
    End Sub
    Public Sub serch_data(ByVal s As String, ByVal r As String)
'查询模块
        If Text_tj.Text.ToString.Trim("") = "" Then
            Response.Write  (" <script  language ='javascript' >
window.confirm('查询内容不能为空！');</script>")
            Text_tj.Focus()
            Exit Sub
        End If
        Select Case DDList_1.Text.ToString.Trim("")
            Case "学号"
                s = "sno"
            Case "姓名"
                s = "sname"
            Case "班级名称"
                s = "class_bj"
            Case "入学年度"
                s = "schoolyear"
        End Select
        Dim sqlcn As New SqlConnection
                sqlcn.ConnectionString = ConfigurationManager.
ConnectionStrings("sqlConnectionString").ConnectionString
        sqlcn.Open()
    Dim strSQL As String = "Select sno,sname,sex,brithday,
nation,schoolyear,classname,class_bj,identity_id,address,
schoolmonth,stuxl,stustate,memory From tablstudent where
" + s + "  = '" + r + "' "
        Dim da As New SqlDataAdapter(strSQL, sqlcn)
```

```
    Dim ds As New DataSet()
    da.Fill(ds, "tablstudent")
    Dim dv As DataView
    dv = ds.Tables(0).DefaultView
    If dv.Count <= 0 Then ' 判断记录数量
       Text_tj.Text = ""
       Text_tj.Focus()
        Response.Write  (" <script  language ='javascript' >
window.confirm(' 没有符合条件的记录！ ');</script>")
    Else
       Details_1.DataSource = dv
       Details_1.DataBind()
       da.Dispose()
       ds.Dispose()
       sqlcn.Close()
       sqlcn.Dispose()
  End If
End Sub
```

查询结果效果如图 4 所示。

图 4 查询结果

3.2 分页功能

由于 DetailsView 控件的数据源采用的是 DataSource 形式，所以除需要将 DetailsView 控件的 AllowPaging 属性设置为 True 之外，还需要手动编写相应代码来处理它的 PageIndexChanging 事件。主要代码如下：

```
Protected Sub Details_1_PageIndexChanging (ByVal sender
As Object, ByVal e As System.Web.UI.WebControls.
DetailsViewPageEventArgs) Handles Details_1.PageIndexCha
nging' 分页功能事件
  If Details_1.CurrentMode = DetailsViewMode.Edit Then
        Details_1.PageIndex = e.NewPageIndex
  Drop_classdata()' 调用 dropdownlist 控件动态添加数据函数
Button_cx_Click(Nothing, Nothing) ' 调用查询按钮的单击事件！
 Details_1.PageIndex = e.NewPageIndex
        Drop_classdata() ' 调用 dropdownlist 控件动态添加
' 数据函数
     End If
If Details_1.CurrentMode = DetailsViewMode.ReadOnly Then
        Details_1.PageIndex = e.NewPageIndex
         Button_cx_Click(Nothing, Nothing) ' 调用查询按钮的
' 单击事件！
     End If
  End Sub
```

3.3 编辑功能

当查询的结果不为空时，单击“编辑”按钮，DetailsView 控件就从默认的 ReadOnly 模式转换为 Edit 模式，具体代码如下：

```
Protected Sub Button_edit_Click (ByVal sender As Object,
ByVal e As System.EventArgs) ' 设置 Detailsview 控件为编辑
' 模式事件
       Details_1.ChangeMode(DetailsViewMode.Edit) ' 设置
'Detailsview 控件为编辑模式
 Button_cx_Click(Nothing, Nothing) ' 调用按钮的单击事件！
 Drop_classdata() ' 调用 DropDownlist 控件动态添加数据函数
End Sub
```

效果图如图 5 所示。

图 5 编辑功能

3.4 更新功能

当修改相关数据之后，就单击“更新”按钮对做出修改的数据项进行保存操作。具体代码如下：

```
Protected Sub Button_updata_Click(ByVal sender As Object,
ByVal e As System.EventArgs) ' 更新数据
     Dim sqlcon As New SqlConnection()
          sqlcon.ConnectionString = ConfigurationManager.
ConnectionStrings("sqlConnectionString").ConnectionString
     Dim strSQL As String
     sqlcon.Open()
      Dim stu_name, stu_sfz, stu_add, stu_tel, stu_bz,
stu_br, stu_snoid As String
stu_snoid = Details_1.DataKey.Value.ToString.Trim ' 取出学号
                    Details_1.Rows (1).Cells (1).FindControl ("
TextBox_stuname").Focus() ' 设置编辑焦点
               stu_name = CType (Details_1.Rows (1).Cells (1).
FindControl ("TextBox_stuname"), TextBox).Text.ToString ().
Trim() ' 取出姓名
```

```
    stu_sfz = CType(Details_1.Rows(8).Cells(1).FindControl("
TextBox_sfz"), TextBox).Text.ToString().Trim() ' 取出身份证号
        stu_add = CType(Details_1.Rows(9).Cells(1).
FindControl("TextBox_addres"), TextBox).Text.ToString().Trim()
' 取出地址
 stu_tel = CType(Details_1.Rows(10).Cells(1).FindControl("
TextBox_tel"), TextBox).Text.ToString().Trim() ' 取出电话
       stu_bz = CType(Details_1.Rows(13).Cells(1).FindControl
("TextBox_bz"), TextBox).Text.ToString().Trim() ' 取出备注
      stu_br = CType(Details_1.Rows(3).Cells(1).FindControl("
TextBox_brday"), TextBox).Text.ToString().Trim() ' 取出生日
            sp = Details_1.Rows(6).Cells(1).FindControl("
DropDownList3") ' 正确获取 Detailsview 控件列模板中的
'DropDownList 控件标示符(专业)
            st = Details_1.Rows(7).Cells(1).FindControl("
DropDownList4") ' 正确获取 Detailsview 控件列模板中的
'DropDownList 控件标示符(班级)
      xl_down = Details_1.Rows(11).Cells(1).FindControl("
DrList2") ' 获得 DropDownList 控件标示符(学历)
      zt_down = Details_1.Rows(12).Cells(1).FindControl("
DropDownList2") ' 获得 DropDownList 控件标示符(状态)
       xb_down = Details_1.Rows(2).Cells(1).FindControl("
DropDownList1") ' 获得性别值
      mz_down = Details_1.Rows(4).Cells(1).FindControl("
DropDList2") ' 获得民族值
 If stu_name.Trim = "" Or stu_sfz.Trim = "" Or stu_add.Trim =
"" Then
   Response.Write("<script language='javascript'>window.
confirm('姓名、身份证号、地址不能为空,请确认!');</script>")
      Else
          strSQL = "update tblstudent set sname ='" &
stu_name & "',sex='" & xb_down.Text.Trim & "',brithday='" &
stu_br & "',nation='" & mz_down.Text.Trim & "',classname='"
& sp.Text.Trim & "' , class_bj ='" & st.Text.Trim & "',
identity_id ='" & stu_sfz & "',address ='" & stu_add & "',
schoolmonth='" & stu_tel & "',stuxl='" & xl_down.Text.Trim
& "',stustate ='" & zt_down.Text.Trim & "',memory ='" &
stu_bz & "' where sno='" & stu_snoid & "'"
   Dim command As New SqlCommand(strSQL, sqlcon)
        command.ExecuteNonQuery()
  End If
     sqlcon.Close()
     Button_cance_Click(Nothing, Nothing) ' 调用取消模块
     End Sub
```

3.5 删除功能

当查询到的数据需要删除时，单击“删除”按钮即可删除数据，具体代码如下：

```
Protected Sub Button_del_Click (ByVal sender As Object,
ByVal e As System.EventArgs) ' 删除数据事件
     Dim snoid As String
          ' 获得当前记录的字段(学号)值的方法:(前提是:给
'Detailsview 控件设置 DataKeynames 属性)
          '1、Details_1.DataKey.Value.ToString.Trim
          '2、Details_1.DataKey.Item(0).ToString.Trim
          snoid = Details_1.DataKey.Value.ToString.Trim ' 获得
' 当前记录的字段(学号)的值
          Dim result As String
          result = MsgBox("确定要删除此学生信息吗? ", 4 + 48,
"信息提示! ")
          If (result = vbYes) Then
              delete_stusno_data(snoid)
    Button_cx_Click(Nothing, Nothing) ' 调用按钮的单击事件!
          Else
              Response.Write ("<script language ='javascript' >
parent.window.history.go(-1);</script>")
          End If
     End Sub
Public Sub delete_stusno_data(ByVal stu_id As String) ' 删除
' 数据函数
          Dim sqlcon As New SqlConnection()
            sqlcon.ConnectionString = ConfigurationManager.
ConnectionStrings("sqlConnectionString").ConnectionString
          Dim strSQL As String
          sqlcon.Open()
strSQL = "delete from tblstudent where sno='" & stu_id & "'"
          Dim command As New SqlCommand(strSQL, sqlcon)
          command.ExecuteNonQuery()
          sqlcon.Close()
     End Sub
```

3.6 取消功能

当对当前的数据修改需要撤销时（没有更新前），单击“取消”按钮即可，具体代码如下：

```
Protected Sub Button_cance_Click (ByVal sender As Object,
ByVal e As System.EventArgs) ' 设置 Detailsview 控件为只读
' 模式事件
     Details_1.ChangeMode(DetailsViewMode.ReadOnly)
' 设置 Detailsview 控件为只读模式
      Button_cx_Click(Nothing, Nothing) ' 调用查询按钮的单
' 击事件!
   End Sub
```

4 结语

结合实例简述了 DetailsView 控件一些基础知识和相关技巧，主要讲解了采用动态数据绑定方法和采用 VB 语言手工编写代码的方法实现数据查询、编辑、更新、删除等操作，在 Visual.Studio.net.2005 和 SQL Server 2000 的环境下测试通过，限于篇幅，本实例的前置及后置代码以源程序为准。

（收稿日期：2013-02-22）

Socket 编程实现局域网远程操作 Office 文档

王文举

摘　要：介绍一种 Socket 编程实现局域网内远程操作 Office 文档的方法及其代码实现。

关键词：Socket 编程；C# 语言；远程操作

1　引言

局域网远程操作一般通过远程桌面或远程控制软件，如果仅需要对远端计算机进行一些简单的操作，用上述方法就太麻烦。文中介绍一种 Socket 编程实现局域网内远程操作 Office 文档的实现方法，可以对远端计算机的 Office 文档（Word，Excel，PPT 等文档）进行翻页操作，其设计思路和实现代码，可供读者借鉴使用。

图 1 服务器端

图 2 客户端

2　设计思路

网络编程几乎都是用 Socket，Socket 是应用层与 TCP/IP 协议族通信的中间软件抽象层，它是一组接口，把复杂的 TCP/IP 协议族隐藏在 Socket 接口后面，由 Socket 去组织数据，以符合指定的协议。服务器端先初始化 Socket，然后与端口绑定（bind），对端口进行监听（listen），调用 accept 阻塞，等待客户端连接。客户端初始化一个 Socket，然后连接到服务器（connect），如果连接成功，客户端发送数据请求，服务器接收请求，并处理请求。如图 1 服务器端（相当于远端计算机）点击“打开文件”，弹出打开文件对话框，选择需要打开的 Office 文档。如图 2 客户端（相当于本地计算机），填写服务器地址，点击“下一页”或“上一页”按键，信息发送给服务器端，服务器端接收到信息，对 Office 文档进行翻页操作。

3　实现代码

编程语言采用 C#，下面介绍图 1 服务器端的实现代码：

```
public partial class Server : Form
{   string host = "127.0.0.1";
    Socket s;
    Socket temp;
    string fname = "";
    string recvStr = "";
    byte[] recvBytes = new byte[1024];
    int bytes;
    public Server()
    { InitializeComponent();}
    private void Server_Load(object sender, EventArgs e)
    {   //系统自动获取本机的 IP 地址
        IPAddress addr = new IPAddress (Dns.
GetHostEntry(Dns.GetHostName()).AddressList[0].Address);
        host = addr.ToString();
        label2.Text = host;
        int port = 2000;//端口号
        IPAddress ip = IPAddress.Parse(host);
        IPEndPoint ipe = new IPEndPoint(ip, port);
            //创建一个 Socket
         s = new Socket (AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
            s.Bind(ipe);//绑定
        s.Listen(10);//开始侦听
    }
    //"打开文件"按键的响应事件
  private void buttonOpen_Click(object sender, EventArgs e)
    {   //选择文件对话框
        OpenFileDialog ofd = new OpenFileDialog();
        ofd.InitialDirectory = "D:\\";
        ofd.Filter = "所有 office 文件|*.**";
        if (ofd.ShowDialog() == DialogResult.OK)
{   fname = ofd.FileName;
    if (fname == "")
    { return;}
}
else
{ return; }
//打开选中的 Office 文档
System.Diagnostics.Process.Start(fname);
do
{   temp = null;
```

```
            bytes = 0;
            recvStr = "";
            //阻塞直到有客户端连接,不然浪费 CPU 资源
                temp = s.Accept();
                //从客户端接收信息
    bytes = temp.Receive(recvBytes, recvBytes.Length, 0);
    recvStr = Encoding.ASCII.GetString(recvBytes, 0, bytes);
//按约定接收到字符 1,向下翻一页,采用响应键盘事件 PGDN
                if (recvStr == "1")
            {  SendKeys.Send("{PGDN}"); SendKeys.Flush(); }
//按约定接收到字符 2,向上翻一页,采用响应键盘事件 PGUP
            if (recvStr == "2")
            {  SendKeys.Send("{PGUP}"); SendKeys.Flush(); }
            temp.Close();
          } while (true);
      }
      //"退出"按键的响应事件
  private void buttonExit_Click(object sender, EventArgs e)
      {  s.Close();  this.Close(); }
}
```

图 2 客户端的实现代码如下：

```
public partial class Client : Form
{  string host = "";
   int port = 2000;
   public Client()
   { InitializeComponent();}
   //"下一页"按键的响应事件
        private void buttonPDown_Click (object sender,
EventArgs e)
      {  if (textBox1.Text == "")
         {  MessageBox.Show("服务器地址不能为空,请填
写! "); return; }
         else
         {  host = textBox1.Text; }
            IPAddress ip = IPAddress.Parse(host);
            IPEndPoint ipe = new IPEndPoint(ip, port);
             Socket c = new Socket (AddressFamily.
InterNetwork, SocketType.Stream, ProtocolType.Tcp); //创建
//一个 Socket
            c.Connect(ipe); //连接到服务器
            string sendStr = "1";
            byte[] bs = Encoding.ASCII.GetBytes(sendStr);
            c.Send(bs, bs.Length, 0); //发送信息
            c.Close();
      }
  }
```

4 结语

通过对一种 Socket 编程实现局域网内远程操作 Office 文档的实现方法的介绍，给出了代码实现。网络编程几乎都是用 Socket，初学者觉得它是比较高深的编程知识，希望此文能抛砖引玉，只要弄清它的工作原理，就揭开了它的神秘面纱。

（收稿日期：2013-01-12）

（上接第 41 页）

```
请选择文件：<input type = "file" name = "file2" size = "30"
maxlength="300"><br>
请选择文件：<input type = "file" name = "file3" size = "30"
maxlength="300">  <input type="submit" value="上传"><br>
请选择文件：<input type = "file" name = "file4" size = "30"
maxlength = "300" >  <input type = "button" value = " 刷新 "
onclick="location.reload();">
</p>
</form>
```

图 7 文件上传页面

代码 13 中，将文件上传桩页面指定为执行模块（第 1 行）；该表单中包含了 4 个类型为文件的输入元素，即可支持批量上传 4 个文件，即上传内容块将会存在 4 个部分（Part）。该页面的展示效果如图 7 所示。

4 结语

全面介绍了 Web 上传组件的应用机制，并以此作为基础，阐述影响 Web 上传组件的通用性和效率的结症所在，并以这些问题点出发，依次提出了改进或规避的方案，特别是将分段处理的缓冲区分为重叠区和更新区的算法，不仅避免了分段对整体信息造成的破坏，又减少了磁盘文件的 I/O，从而巧妙地实现了该组件在多文件批量上传和大文件上传等方面存在不通用和效率不高的问题。

相比后台功能的接触深度，前端的用户体验度还有待加强，例如：上传状态的异步更新、传输状态的异步监测等，都将是值得后期改进的内容。

（收稿日期：2013-01-20）

ASP 的网络聊天室设计与开发

杨 东

摘 要：讲述了使用 ASP 设计与开发网络聊天室的方法和过程。通过使用 Application 对象，记录和显示了用户的聊天信息，利用不同的分隔符，将用户发送的聊天信息进行整合并存储在 Application 变量中，在用户接收时加以解释和显示。用户在聊天室中可以查看在线的用户列表，可以发送公共消息，也可以发送私人消息。

关键词：网络聊天室；ASP 技术；Application 对象；聊天信息

1 引言

互联网的飞速发展给人们的生活带来了巨大的变化，网络聊天已成为诸多网络用户日常生活的一部分，用户可通过 QQ、MSN 等即时通信工具软件进行聊天，也可通过在线的网络聊天室进行交流。将讲述网络聊天室设计和开发过程。

2 设计思想

用户通过首页登录聊天室之后，系统向所有在线用户发出新用户登录提醒。用户能看到系统或者是其他用户发布的公共消息，还能够与在线的用户进行一对一的私人交流。聊天信息中可加入表情动画，表现说话人的情感。

聊天室主界面采用上中下结构的框架式设计，顶部含 Logo 等信息，中间分为两部分，左边显示聊天信息，右边显示在线用户列表，底部为消息发送窗口。聊天内容和在线用户显示窗口均采用定时刷新的方式，以更新内容。设计结构如图 1 所示。

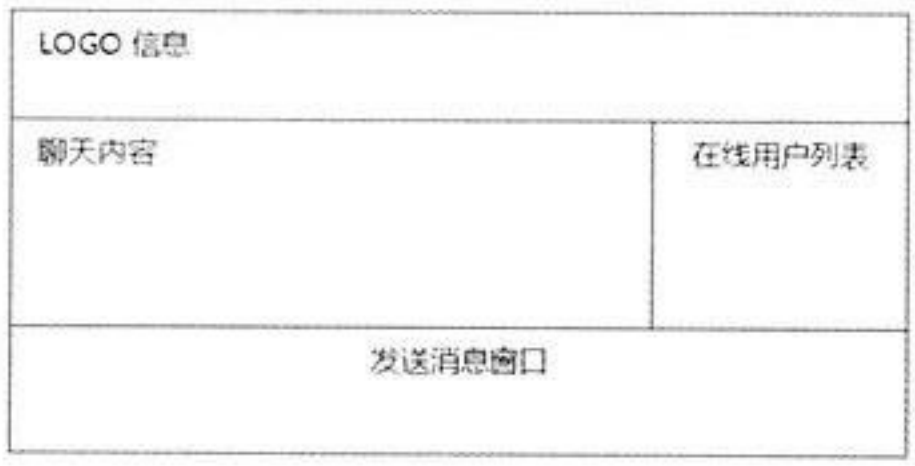


图 1 聊天室结构

3 数据库

考虑到聊天内容信息量大，读取信息的操作频繁，若将聊天内容先存入数据库，再行读取，将会大大延迟系统的执行效率，造成数据传输延时严重等影响，因而聊天内容使用 Application 对象来存储，后文详细介绍。

数据库“user.mdb”仅包含一个表“tbUser”，其中存储了用户的信息，以便用户进行注册操作和登录时的身份验证。“tbUser”表的结构如表 1 所示。

表 1 tbUser 表结构

字段名	类型	说明
ID	自动编号	自动编号
UserName	文本	用户名
UserPwd	文本	用户密码
UserSex	文本	用户的性别

4 聊天室开发

4.1 数据库连接

当存在多个页面都需要连接数据库的情况时，可将数据库连接代码编写成公用文件，此方法将大大减少代码的繁杂度，增加代码的可修改性。本系统将数据库连接代码放入公用文件“conn.asp”中，在需要连接数据库的时候，在文件中添加包含<! --#include file=“conn.asp” -->即可。conn.asp 的代码如下：

```
<%
'数据库连接
Dim conn
Set conn= Server.CreateObject("ADODB.Connection")
conn.Open "Driver={Microsoft Access Driver  (*.mdb)};Dbq="
&Server.MapPath("user.mdb")
%>
```

4.2 Global.asa 文件

在本系统中使用 Application 对象来记录用户的聊天内容，

因此需要使用“Global.asa”文件来记录程序的启动、会话的开始与结束，进行相应的变量初始化，在用户登录时进行信息处理。具体代码如下：

```
<Script Language ="VBScript" RunAt="Server">
'程序开始
Sub Application_OnStart
    '记录用户列表
    Dim Online(100,2)
    '记录聊天室的所有聊天信息
    Dim TempArray(50)
    Application("Online")=Online
    Application("Talk")=TempArray
    Application("TalkIndex")=0
End Sub
'会话开始
Sub Session_OnStart
    Session("Start")=Now
    '刷新时间设置
    Session("Refresh")="5"
End Sub
'会话结束
Sub Session_OnEnd
    '结束时将用户从列表中删除
    Dim tempOnline,i
    Application.Lock
    tempOnline=Application("Online")
    For i=0 To UBound(TempOnline)
        If tempOnline(i,0)=Session("name") Then
            tempOnline(i,0)=""
            Exit For
        End If
    Next
    Application("Online")=tempOnline
  Application.Unlock
End Sub
</Script>
```

4.3 聊天室首页

首页“index.asp”为用户提供了登录的入口，新用户也可通过此页面的链接进行账号注册。如图 2 所示。

图 2 聊天室首页

该部分代码如下：

```
<html>
<body>
<h2 align = "center">欢迎登录聊天系统</h2>
<form method="post" action="loginJump.asp">
<br> <table align="center">
        <tr>
            <td>用户名:</td>
        <td><input type ="text" name = "txtName"></td>
        </tr>
        <tr>
            <td>密　码:</td>
      <td><input type="password" name= "txtPwd"></td>
        </tr>
        <tr>
            <td></td>
            <td ><br ><input  type = "submit"  name = "
btnSubmit" value="登录"
style="width:50px;height:25px"> 
      <input type="reset" name="btnReset" value="重置"
style = "width:50px;height:25px" >  <a  href = "register.asp" > 注
册</a></td>
        </tr>
</form>
</body>
</html>
```

用户通过个人账号和密码登录后，转入“loginJump.asp”页面处理，验证用户输入的账号信息。用户通过信息验证后，将用户信息和消息记录保存在 Application 变量中。该部分代码如下：

```
<! --#include file="conn.asp"-->
<%
userName=Request.form("txtName")
userPwd=Request.form("txtPwd")
If userName="" Or userPwd="" Then
    response.Write("<script> alert(' 用户名或者密码不能为
空,请重新登录！ ');location=('index.asp') </script>")
Else
' 判断用户名是否存在
    sqlExist = "select  *  from  tbUser  where  UserName ='"
&userName&"'"
    Set rsExist=Server.CreateObject("ADODB.Recordset")
    rsExist.open sqlExist,conn,1,1
    If rsExist.eof Then
        response.Write("<script> alert(' 用户名不存在，请
先注册！ ');location=('register.asp') </script>")
    else
        sql = "select  *  from  tbUser  where  UserName ='"
&userName&"' and UserPwd='"&userPwd&"'"
        Set rs=Server.CreateObject("ADODB.Recordset")
```

```
        rs.open sql,conn,1,1
        If rs.eof Then
            response.Write("<script> alert('密码与用户名
不匹配,请重新登录!');location=('index.asp') </script>")
        Else
            '记录用户身份
            Session("name")=userName
            Session("sex")=rsExist("UserSex")

            Application.Lock
                tempOnline=Application("Online")
            Application.UnLock
            '判断用户名是否已经登录,若已登录返回错误
'信息并跳转至首页
            For i=0 To UBound(tempOnline)
        If tempOnline(i,0)=Session("name") Then
        response.Write("<script> alert('该用户名已登录,
不可重复登录!');location=('index.asp') </script>")
                    Response.End
                End If
            Next
            Application.Lock
                '用户名并未登录,保存用户信息
                For j=0 To UBound(tempOnline)
                    '找到数组尾部
                    If tempOnline(j,0)="" Then
                        '记录用户名
                tempOnline(j,0)=Session("name")
                        '记录用户性别图片
                        tempOnline (j,1)="<img src=
images/"&Session("sex")&".gif>"
                        '保存后跳出循环
                        Exit For
                    End If
                Next
                '记录在线列表
                Application("Online")=tempOnline
                '记录聊天内容
                strTalk = " <xitong | 所 有 人 @wu $ "
&Session("name")&"进入聊天室" &"   "&Time()&">"
                tempTalk=Application("Talk")
        tempTalk(Application("TalkIndex")+1)=strTalk
        Application("TalkIndex")=Application("TalkIndex")+1
                Application("Talk")=tempTalk
    Application(Application("TalkIndex"))=tempArray
                '用户人数加 1
        Application("TalkIndex")=Application("TalkIndex")+1
            Application.UnLock
            Response.redirect "main.asp"
        End if
    End if
End If
conn.close
%>
```

4.4 用户注册页面

新用户点击“注册”链接进入注册页面“register.asp”，输入相关信息即可进行注册。如图 3 所示。

图 3 用户注册

部分代码如下：

```
<html>
<body>
<h2 align = "center">欢迎注册聊天系统</h2>
<form method="post" action="insUser.asp">
<br> <table align="center">
        <tr>
            <td>请输入用户名:</td>
        <td><input type ="text" name = "txtName"></td>
        </tr>
        <tr>
            <td>请选择性别:</td>
            <td ><input type ="radio" name = "rdoSex"
value="男" checked>男
        <input type ="radio" name = "rdoSex" value="女">女
            </td>
        </tr>
        <tr>
            <td>请输入密码:</td>
        <td><input type="password" name= "txtPwd1"></td>
        </tr>
        <tr>
            <td>请确认密码:</td>
            <td ><input  type = "password"  name = "
txtPwd2"></td>
        </tr>
        <tr>
            <td></td>
            <td ><br ><input  type = "submit"  name = "
btnSubmit" value="注册"
style="width:50px;height:25px"> 
                        <input  type = "reset"  name = "
btnReset" value="重置"
style="width:50px;height:25px"> <a href ="index.asp">返回</
a></td>
```

```
        </tr>
</form>
</body>
</html>
```

用户填写信息后，点击“注册”按钮，进入注册处理页面“insUser.asp”，将用户信息记入数据库中。该部分代码如下：

```
<! --#include file="conn.asp"-->
<%
If Request.form("txtName")="" Or Request.form("txtPwd1")=""
Or Request.form("txtPwd1")="" Then
    response.Write("<script> alert('请将信息填写完整！');
location=('register.asp') </script>")
Else
    sqlExist = "select * from tbUser where UserName ='"
&Request.form("txtName")&"'"
    Set rsExist=Server.CreateObject("ADODB.Recordset")
    rsExist.open sqlExist,conn,1,1
    '用户名是否存在
    If rsExist.eof Then
        '输入的密码不一致
        If Request.form("txtPwd1") <>Request.form("
txtPwd2") Then
            response.Write("<script> alert('两次输入的密
码不一致！');location=('register.asp') </script>")
        Else
            '插入信息至数据库
            sql="insert into tbUser(UserName,UserPwd,
UserSex) values ('"&Request.form("txtName")&"','"&Request.
form("txtPwd1")&"','"&Request.form("rdoSex")&"') "
            conn.Execute(sql)
            response.Write("<script> alert('注册成功，点
击跳转至登录页面！');location=('index.asp') </script>")
        End if
    Else
        response.Write("<script> alert('用户名已存在，请
重新填写！');location=('register.asp') </script>")
    End If
End If
conn.close
%>
```

4.5 聊天室主页

“main.asp”页面为聊天室的主页面，该框架页面包含了“top.asp”、“chatContent.asp”、“userList.asp”、“sendMsg.asp”4个子页面。页面效果如图4所示。

“main.asp”页面该部分代码如下：

```
<html>
<head>
<title>聊天系统</title>
</head>
    <frameset rows="15%,*,20%">
        <frame name="top" src="top.asp">
        <frame name="middle" src="middle.asp">
        <frame name="bottom" src="message.asp">
        </frameset>
</html>
```

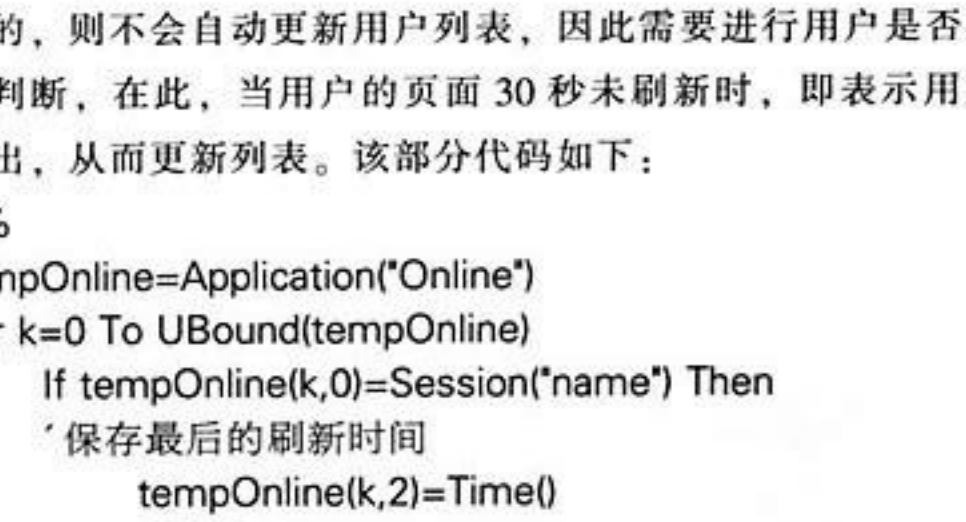


图4 聊天室主页面

4.6 在线用户列表

在线用户列表页面“userList.asp”每10秒自动刷新一次，动态更新在线的用户信息。用户可通过其中的“退出”链接进行退出，并清除相关记录内容。但若用户是通过关闭浏览器退出的，则不会自动更新用户列表，因此需要进行用户是否在线的判断，在此，当用户的页面30秒未刷新时，即表示用户已退出，从而更新列表。该部分代码如下：

```
<%
tempOnline=Application("Online")
For k=0 To UBound(tempOnline)
    If tempOnline(k,0)=Session("name") Then
    '保存最后的刷新时间
        tempOnline(k,2)=Time()
        Exit For
    End If
Next
For k=0 To UBound(tempOnline)
    If tempOnline(k,0)<>"" And tempOnline(k,2)<>"" Then
    '对已记录时间的用户进行判断
        If DateDiff("s",tempOnline(k,2),Time())>20 Then
        '30秒未刷新则表示用户退出
            tempOnline(k,0)=""
            tempOnline(k,1)=""
            tempOnline(k,2)=""
        End If
    End if
Next
Application.Lock
    Application("Online")=tempOnline
Application.UnLock
%>
<html>
<head>
    <meta http-equiv="Refresh" content="10">
```

```
    <title>用户在线列表</title>
</head>
<body>
<center><br><font size="2" color="Green"><b>在线用户</
b></font><br><br>
<table>
<%
Application.Lock
    tempOnline=Application("Online")
Application.UnLock

For i=0 To UBound(tempOnline)
    If tempOnline(i,0)<>"" Then
    ' 读取在线用户列表(头像和用户名)
        If tempOnline(i,0)=Session("name") Then
%>
        <tr bgcolor="fff000">
            <td><%=tempOnline(i,1)%></td>
            <td><%=tempOnline(i,0)%></td>
        </tr>
<%
        Else
%>
        <tr>
            <td><%=tempOnline(i,1)%></td>
            <td><%=tempOnline(i,0)%></td>
        </tr>
<%
        End If
    End If
Next
%>
</table>
<p><a href="logOut.asp" onclick="return confirm (' 您确定要
退出？ ')" target="_top">退出</a>
</body>
</html>
```

4.7 发送消息

在页面“message.asp”中，用户可选择表情，输入消息内容，并选择消息的发送对象。如图 5 所示。

图 5 发送消息窗口

代码如下：

```
<html>
<body>
    <form method="POST" action="sendMsg.asp">
        <table width="60%" border="0" align="center">
            <tr>
                <td align="center" valign="middle">
                    <select  name = "sltBiaoQing"
style="width:80px;height:25px;font-size:14px">
                        <option value="wu">无</option>
                        <option value="aoman">傲慢</option>
                        <option value="baiyan">白眼</option>
                        <option value="bishi">鄙视</option>
                        <option value="ciya">呲牙</option>
                        <option value="dabing">大兵</option>
                        <option value="deyi">得意</option>
                        <option value="fanu">发怒</option>
                        <option value="ganga">尴尬</option>
                        <option value="haqian">哈欠</option>
                        <option value="huaixiao">坏笑</option>
                        <option value="jie">饥饿</option>
                        <option value="jingya">惊讶</option>
                        <option value="keai">可爱</option>
                        <option value="koubi">口鼻</option>
                        <option value="lenghan">冷汗</option>
                        <option value="piezui">撇嘴</option>
                        <option value="tiaopi">调皮</option>
                        <option value="touxiao">偷笑</option>
                        <option value="weixiao">微笑</option>
                        <option value="youhengheng">右哼哼</option>
                        <option value="zuohengheng">左哼哼</option>
                        <option value="zaijian">再见</option>
                    </select>
                </td>
                <td align="center" valign="middle">
<textarea name="txtMsg" rows="3" cols="60"></textarea>
                </td>
                <td align="center" valign="middle">
                    <select  name = "sltReceiver"
style="width:80px;height:25px">
                        <option value="所有人">所有人</option>
                        <%
                        tempOnline=Application("Online")
                        For i=0 To UBound(tempOnline)
                            If tempOnline (i,0) <> ""
And tempOnline(i,0)<>Session("name") Then
                            %>
                            <option  value = "
<%=tempOnline(i,0)%>"><%=tempOnline(i,0)%></option>
                            <%
                            End If
                        Next
                        %>
                    </select>
                </td>
                <td align="center" valign="middle">
                    <input  type = "submit"  name = "
btnSubmit" value="发送" style="width:80px;height:40px">
                    <input  type = "reset"  name = "
```

```
btnReset" value="清空"  style="width:50px;height:40px">
                    </td>
                </tr>
            </table>
</body>
</html>
```

"sendMsg.asp" 页面对用户发送的信息进行处理。将单条消息按照特定的格式封装，并加入到 Application 对象的数组变量 "Talk" 中，在聊天内容显示页面再行显示。该部分代码如下：

```
<%
If Request.form("txtMsg")<>"" Then
    ´一次只能显示 50 条记录,多了则从头重新开始
    If Application("TalkIndex")>49 Then
        Application("TalkIndex")=0
    End If
    ´消息格式<发送人|接收人 @ 表情$消息内容>
    strTalk = " < "&Session  ("name")&" | "&Request.form ("
sltReceiver")&"@"&Request.form("sltBiaoQing")&"$"&Request.
form("txtMsg")&"   "&Time()&">"
    Application.Lock
        tempTalk=Application("Talk")
        tempTalk(Application("TalkIndex")+1)=strTalk
        Application("TalkIndex")=Application("TalkIndex")+1
        Application("Talk")=tempTalk
    Application.UnLock
    Response.Redirect "message.asp"
Else
    Response.Redirect "message.asp"
End if
%>
```

4.8 聊天内容显示

"chatContent.asp" 页面显示了即时的聊天记录，并设置了 5 秒自动刷新一次，以更新消息。对于所有聊天记录，选择接收对象为 "所有人" 或者用户名为当前登录者的人（私人的消息）显示。该部分代码如下：

```
<html>
<head>
    <meta http-equiv="Refresh" Content="<%=Session("
Refresh")%>">
</head>
<body>
<table width="100%" border="0" align="center">
<%
tempTalk=Application("Talk")
For i=Application("TalkIndex") To 0 Step -1
    ´不同行颜色交叉
    If i Mod 2=0 Then
        lineColor="#ffffff"
    Else
        lineColor="#efefef"
    End If
    If tempTalk(i)<>"" Then
    ´消息不空。分离出说话人,接收人,说话表情,说话的内容
        strSender =Mid (tempTalk (i),InStrRev (tempTalk (i),"
<")+1,InStrRev(tempTalk(i),"|")-InStrRev(tempTalk(i),"<")-1)
        strReceiver=Mid(tempTalk(i),InStrRev(tempTalk(i),"
|")+1,InStrRev(tempTalk(i),"@")-InStrRev(tempTalk(i),"|")-1)
        strBiaoQing=Mid(tempTalk(i),InStrRev(tempTalk(i),"
@")+1,InStrRev(tempTalk(i),"$")-InStrRev(tempTalk(i),"@")-1)
        strMsg=Mid(tempTalk(i),InStrRev(tempTalk(i),"$")+
1,InStrRev(tempTalk(i),">")-InStrRev(tempTalk(i),"$")-1)
        If strReceiver<>"所有人" Then
        ´私人消息
            If  strReceiver=Session("name") Then
%>
                <tr>
                    <td bgcolor = "<% =lineColor% > "
><%=strSender%>对我说：<%If strBiaoQing<>"wu" Then %
><img src="biaoqing/<%=strBiaoQing%>.gif"><% End If %
><%=strMsg%></td>
                <tr>
<%
            ElseIf strSender=Session("name") Then
%>
                <tr>
                    <td bgcolor="<%=lineColor%>">
我对<%=strReceiver%>说：<%If strBiaoQing<>"wu" Then
%><img src="biaoqing/<%=strBiaoQing%>.gif"><% End If
%><%=strMsg%></td>
                <tr>
<%
            End If
        Else
        ´不是私聊
%>
<%
        If strSender="xitong" Then ´是系统发送的消息
%>
                <tr>
                    <td bgcolor="<%=lineColor%>">
系统说：<img src="images/enter.gif"><%=strMsg%></td>
                <tr>
<%
            ElseIf strSender=Session("name") Then ´是自
´己发送的消息
%>
                <tr>
                    <td bgcolor="<%=lineColor%>">
我对所有人说：<%If strBiaoQing<>"wu" Then %><img src="
biaoqing/<% =strBiaoQing% > .gif" ><% End If % ><% =
strMsg%></td>
```

（下转第 74 页）

用 C# 实现 TFTP 协议及其应用

明廷堂

摘　要：全面解读了 TFTP 协议的技术细节，并在此基础上运用 C# 语言完整实现该协议。通过一个 TFTP 客户端、服务器应用，在真实的网络管理环境中进行测试。项目保持完整性、独立性，为 Socket 编程开发基于网络协议的应用程序提供了一个基本思维框架。

关键词：简单文件传输协议；客户端应用；服务端应用；网络管理

1　TFTP 概述

TFTP 是一个基于 UDP 的简单文件传输协议。其设计初衷是进行小文件传输的，因此与 FTP 相比，它只能从文件服务器上获得或写入文件，不能列出目录，不进行认证。它传输 8 位数据，有 3 种传输模式：netascii，特殊的 8 位 ASCII 码；octet，8 比特位组数据类型；mail，目前已不再支持，它将返回的数据直接返回给用户而不是保存为文件。TFTP 的优点在于实现简单而不是系统的高吞吐量。

值得注意的是协议端口：TFTP 协议需要客户进程向服务器进程的熟知 UDP 端口 69 发送第一个分组，服务器进程监听到此请求后，就向服务器主机申请一个尚未使用的临时端口，然后服务器进程使用该临时端口与该请求的客户进程进行数据交换。这种端口切换原因是：服务器进程不能长期占用这个熟知端口来完成一些需要较长时间 (可能是几十秒或数分钟) 的文件传输，在传输当前文件的过程中，这个熟知端口要留出来供服务器进程监听其他的 TFTP 客户进程发送的并发请求。

2　TFTP 报文及其选项

一个 TFTP 分组包含几个子域：本地媒介头，IP 头，数据报头，TFTP 头，剩下的就是 TFTP 数据了。TFTP 在 IP 头中不指定任何数据，但它使用 UDP 中的源和目标端口以及包长度域。TFTP 使用的分组标记 (TID) 在连接初始化时用作 UDP 端口，且必须介于 0 到 65,535 之间。TFTP 分组的头部顺序如图 1 所示。

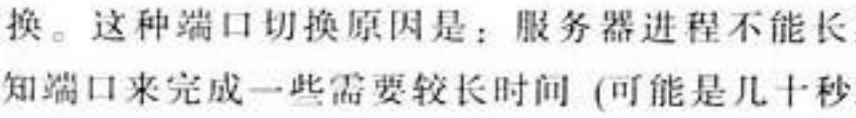


图 1　TFTP 分组的头部顺序

2.1　RRQ/WRQ 报文

TFTP 的读请求报文 (RRQ) 和写请求报文 (WRQ) 具有相同的格式。如图 2 所示。

图 2　TFTP 的 RRQ/WRQ 报文格式

每个 TFTP 报文都包含一个 2 字节的操作码。RRQ、WRQ 报文的操作码分别为 0x01、0x02。文件名指定客户端要读取或写入的文件服务器上的文件。文件名字段以 1 字节长的 0 字节作为结束符。模式字段是一个 ASCII 码串字符 netascii 或 octet (大小写可任意组合)，其中：netascii 表示数据是以成行的 ASCII 码字符组成，以两个字节的回车字符后跟换行字符 (简称 CR/LF) 作为行结束符。CR/LF 在这种格式和本地主机使用的行定界符之间进行转化；octet 则表示将数据看作 8 比特位组的字节流而不作任何解释。模式字段同样以一个字节长的 0 字节结束。

2.2　DATA 报文

TFTP 的数据报文 (DATA) 用于实际数据的传输。如图 3 所示。

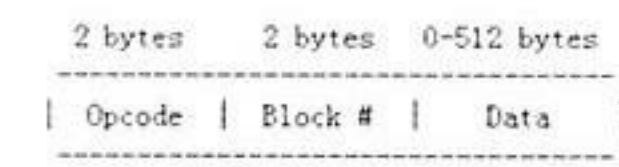


图 3　TFTP 的 DATA 报文格式

DATA 报文的操作码为 0x03。每个 DATA 分组包含一个 2 个字节的块编号字段，用于后续传输过程的数据块标识和确认。数据域从 0 字节到 512 字节。如果数据域是 512 字节，则指示延续传输过程，如果小于 512 字节则表示这是最后一个分组，传输终止。

2.3　ACK 报文

TFTP 的确认报文 (ACK) 格式如图 4 所示。

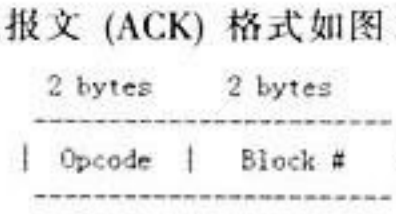


图 4　TFTP 的 ACK 报文格式

ACK 报文的操作码为 0x04。块编码域是前一次传输的数据块的编码，用于确认该数据块是否成功传输。

2.4 ERROR 报文

TFTP 的差错报文 (ERROR) 格式如图 5 所示。

2 bytes	2 bytes	string	1 byte
Opcode	ErrorCode	ErrMsg	0

图 5 TFTP 的 ERROR 报文格式

ACK 报文的操作码为 0x05。如果请求不能被满足，或者在传输中发生错误，需要发送 ERROR 包。它用于 TFTP 服务器不能处理读请求或写请求的情况。在文件传输过程中的读和写如果出现差错也会导致发送这种报文，并停止传输。差错报文的差错编号字段给出一个差错码 (2 字节长)，后面是一个用 ASCII 码表示的差错报文字段。由于 TFTP 报文使用不可靠的 UDP 进行传输，TFTP 就必须处理分组丢失和分组重复。分组丢失可通过发送方的超时与重传机制解决。同其他的 UDP 应用一样，TFTP 报文中没有校验和，它假定任何数据差错都将被 UDP 的检验和感知。

2.5 TFTP 选项

RFC2347 描述了 TFTP 的选项扩展机制。

RFC2348 描述了一个 TFTP 扩展选项 blksize。该选项允许客户端和服务器协商数据报文的块大小以更有效适应当前网络媒体 (其 MTU 已经达到 1500)，从而提高传输效率。如果 TFTP 服务器接受 blksize 选项，它必须向客户端发送一个 OACK 报文回应 blksize 值，服务器指定的 blksize 值不能超过客户端指定的 blksize 的值。客户端或者使用 OACK 中指定的 blksize 值或者发送错误代码为 8 的 ERROR 报文终止传输。

RFC2349 描述了 TFTP 的两个扩展选项:tsize 与 timeout。tsize 扩展允许在读请求或写请求中指定请求的文件的大小，并由接收文件的一方最终决定传输窗口的尺寸。对于读请求，RRQ 报文在 tsize 域指定 “tsize”，在 size 域指定 “0”，文件的大小 (8 位位组形式) 在 OACK 报文中返回。如果文件太大以至于客户端不能处理，客户端将产生一个错误代码为 3 的异常并终止传输；对于写请求，WRQ 报文在 tsize 域指定 “tsize”，在 size 域指定文件的大小，文件的大小 (八位位组形式) 在 OACK (Option ACKnowledgment) 报文中回应。如果文件太大以至于服务器不能处理，服务器将产生一个错误代码为 3 的异常并终止传输。timeout 扩展选项允许 TFTP 服务器和客户端在读请求或写请求中指定请求超时间隔。如果 TFTP 服务器允许接受超时选项，它向客户端发送一个 OACK 报文，其指定的超时值必须同客户端指定的超时值相匹配。

3 TFTP 交互过程

TFTP 使用简单的停止-等待协议进行数据传输。

在读操作下，TFTP 客户端首先需要向服务器发送一个 RRQ 说明要读取的文件名和文件模式。如果该文件能被该客户读取，TFTP 服务器就返回一个块编为 1 的数据分组。TFTP 客户端成功接收后发送一个块编号为 1 的 ACK。TFTP 服务器随后发送块编号为 2 的数据分组。TFTP 客户端成功接收后发送一个块编号为 2 的 ACK。如此重复，直至该文件传送完成。除了最后一个数据分组可包含有不足 512 字节的数据，其他每个数据分组均含有 512 字节的数据。当 TFTP 客户端收到一个不足 512 字节的数据分组，就知道它收到最后一个数据分组。

写操作与读操作类似，TFTP 客户端发送 WRQ 指明文件名和模式。如果该文件能被该客户写入，TFTP 服务器就返回块编号为 0 的 ACK 报文。该客户端就将文件的前 512 字节以块编号为 1 发出。服务器就返回块编号为 1 的 ACK 报文。如此反复，直至文件传送完成。

图 6 描述了详细的 TFTP 交互过程。

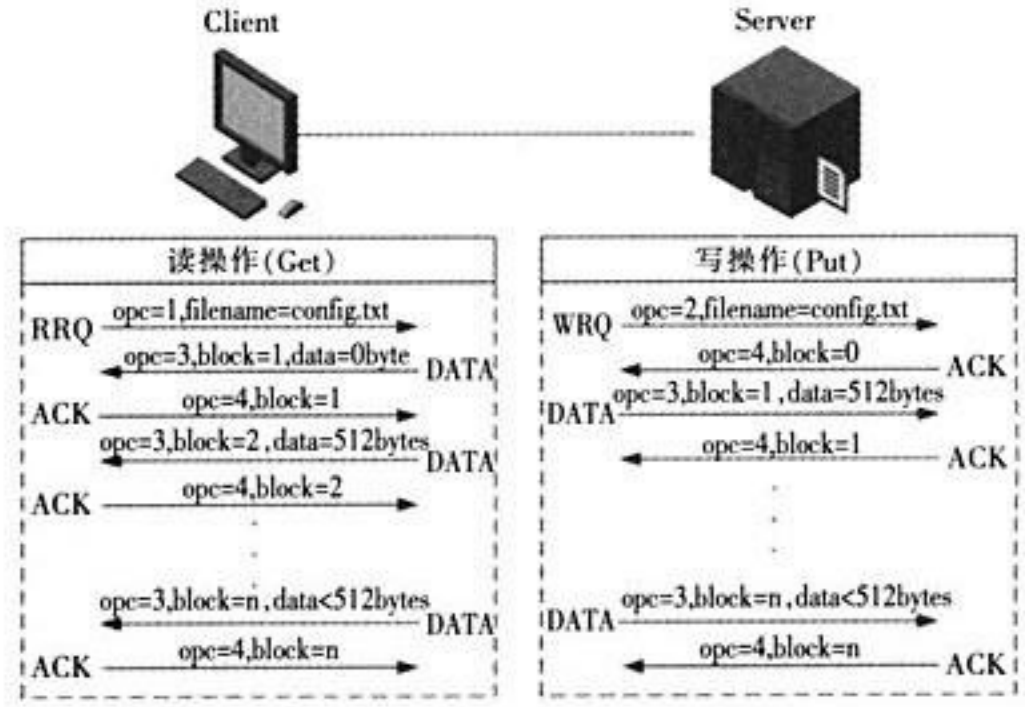


图 6 TFTP 数据交互过程

4 TFTP 的 C# 完整实现

TFTP 的 C# 完整实现是一个整体架构：TFTPLibrary 实现整个 TFTP 协议；TFTPClient 是一个客户端应用程序；TFTPServer 是一个服务器应用程序。架构如图 7 所示。

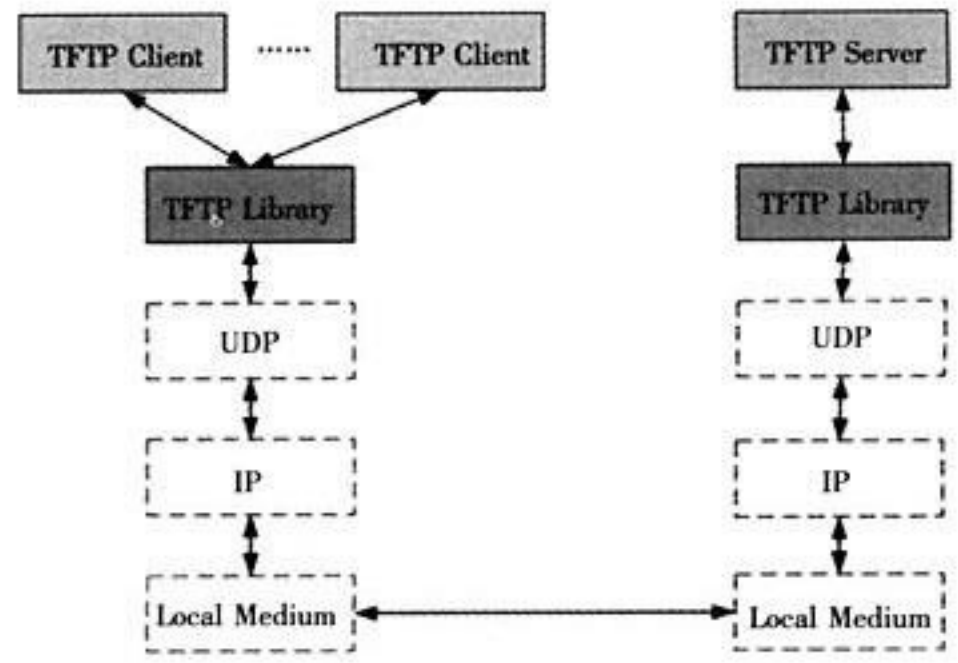


图 7 TFTP 完整实现的整体架构

4.1 TFTPLibrary 实现过程

TFTPLibrary 包含 4 个主要部分：事件参数设置模块、单线程报文处理模块、多线程报文处理管理模块、传输会话管理模块。

单线程报文处理程序 (TFTPProcess) 是核心模块，它是传输任务的真正执行者。TFTPProcess 定义了一些方法，一方面用于 TFTP 报文的构造、发送与接收，另一方面用于传输会话 (TFTPSession) 的管理。

任何网络应用程序离不开底层的数据发送、接收。下面的源代码采用 Socket 编程以及回调的方式实现异步发送和接收数据：

```
privatevoid SendCallback(IAsyncResult result)
{
try
   {
      ((Socket)result.AsyncState).EndSendTo(result);
   }
catch (ObjectDisposedException)
   { }
}
privatevoid Send(IPEndPoint RemoteEndPoint, byte[] Data)
{
try
   {
         LocalSocket.BeginSendTo (Data, 0, Data.Length,
SocketFlags.None, RemoteEndPoint,
newAsyncCallback(SendCallback),LocalSocket);
   }
catch (Exception ex)
   { }
}
privatevoid Receive()
{
while (Loop)
   {
bool ReceivedWorked = false; Byte[] ReceiveBuf = newByte
[MaxRecvSize];
EndPoint tempPoint = newIPEndPoint(IPAddress.Any, 0); int
NumBytesReceived = 0;
try
      {
            NumBytesReceived = LocalSocket.ReceiveFrom
(ReceiveBuf, ref tempPoint);
         ReceivedWorked = true;
      }
catch (Exception ex)
      {
         if (ex.GetType().FullName == "System.Net.Sockets.
SocketException"&&
            ((SocketException)ex).ErrorCode ! = 10004)
         {
this.StopListener();
         }
      }
if (ReceivedWorked)
      {
   IPEndPoint RemoteEndPoint=(IPEndPoint)tempPoint; bool
MatchIP = true;
if (ClientMode)
         {
            MatchIP = false; bool blsIPStr= false;
try
            {
IPAddress HostAddr = IPAddress.Parse(ClientHost);
               blsIPStr = true;
if  (RemoteEndPoint.Address.Equals  (HostAddr)) MatchIP =
true;
            }
catch
            {}
if (blsIPStr == false)
            {
foreach (IPAddress ip inDns.GetHostAddresses(ClientHost))
               {
if (ip == RemoteEndPoint.Address)  MatchIP = true;
               }
            }
         }
if (MatchIP)
         {
      ProcessDatagram   (ReceiveBuf,   NumBytesReceived,
RemoteEndPoint);
         }
      }
   }
}
```

从 Socket 缓冲区接收到 TFTP 报文后，就需要调用 ProcessDatagram () 方法对报文进行处理，而 ProcessDatagram() 又调用各种针对具体 TFTP 报文类型的处理方法。下面简要描述这些处理方法。

TFTP 报文处理的一项重要工作是根据接收到的确认报文 (ACK) 的块编号发送下一个 DATA 报文，下面的源代码片段实现了这一处理过程：

```
privatevoid SendNextDataDatagram (byte [] ReceivedBytes,
IPEndPoint RemoteEndPoint, string EndPointString)
{
try
   {
     byte [] DataBytes = session.GetData (ReceivedBytes[2],
```

```
ReceivedBytes[3]);
byte[] SendBytes = newbyte[DataBytes.Length + 4];
        DataBytes.CopyTo(SendBytes, 4);
        SendBytes[1] = 3;
        SendBytes[2] = session.BlockIDByte1;
        SendBytes[3] = session.BlockIDByte2;
         LogMsg (Level.Verbose, GlobalID.ToString () + ":
Sending bytes to " + EndPointString);
        Send(RemoteEndPoint, SendBytes);
    }
catch (System.IO.IOException ex)
    {
        Send(RemoteEndPoint, Error1);
        session.Close();
        StopListener();
    }
}
```

TFTP 的终极目的是实现文件的传输。从客户端的角度，对于上传操作，需要调用 PutFile () 方法；对于下载操作，需要调用 GetFile () 方法。下面的源代码片段实现了 GetFile () 文件交互过程：

```
publicvoid GetFile (string Host, int Port, string Filename, bool
Filesize, int Timeout, int BlockSize)
{
    ClientMode = true;
    ClientHost = Host;
    IPEndPoint RemoteEndPoint = newIPEndPoint(IPAddress.
Parse(Host), Port);
    #region Construt Standard RRQ Datagram
ArrayList DataBytes = newArrayList(20);
    DataBytes.Add(Convert.ToByte(0));
    DataBytes.Add(Convert.ToByte(1));
      byte[] filebytes = System.Text.Encoding.ASCII.GetBytes
(Filename);
foreach (byte FileByte in filebytes)
    {
        DataBytes.Add(FileByte);
    }
    DataBytes.Add(Convert.ToByte(0));
    DataBytes.Add(Convert.ToByte(Convert.ToChar("o")));
    DataBytes.Add(Convert.ToByte(Convert.ToChar("c")));
    DataBytes.Add(Convert.ToByte(Convert.ToChar("t")));
    DataBytes.Add(Convert.ToByte(Convert.ToChar("e")));
    DataBytes.Add(Convert.ToByte(Convert.ToChar("t")));
    DataBytes.Add(Convert.ToByte(0));
    #endregion
    #region Construct RRQ Options
if (Filesize)
    {
        DataBytes.Add(Convert.ToByte(Convert.ToChar("t")));
        DataBytes.Add(Convert.ToByte(Convert.ToChar("s")));
        DataBytes.Add(Convert.ToByte(Convert.ToChar("i")));
        DataBytes.Add(Convert.ToByte(Convert.ToChar("z")));
        DataBytes.Add(Convert.ToByte(Convert.ToChar("e")));
        DataBytes.Add(Convert.ToByte(0));
        DataBytes.Add(Convert.ToByte(Convert.ToChar("0")));
        DataBytes.Add(Convert.ToByte(0));
    }
if (BlockSize ! = 512)
    {
        DataBytes.Add(Convert.ToByte(Convert.ToChar("b")));
        DataBytes.Add(Convert.ToByte(Convert.ToChar("l")));
        DataBytes.Add(Convert.ToByte(Convert.ToChar("k")));
        DataBytes.Add(Convert.ToByte(Convert.ToChar("s")));
        DataBytes.Add(Convert.ToByte(Convert.ToChar("i")));
        DataBytes.Add(Convert.ToByte(Convert.ToChar("z")));
        DataBytes.Add(Convert.ToByte(Convert.ToChar("e")));
        DataBytes.Add(Convert.ToByte(0));
string size = BlockSize.ToString();
foreach (char ch in size)
            DataBytes.Add(Convert.ToByte(ch));
        DataBytes.Add(Convert.ToByte(0));
    }
if (Timeout ! = 1)
    {
        DataBytes.Add(Convert.ToByte(Convert.ToChar("t")));
        DataBytes.Add(Convert.ToByte(Convert.ToChar("i")));
        DataBytes.Add(Convert.ToByte(Convert.ToChar("m")));
        DataBytes.Add(Convert.ToByte(Convert.ToChar("e")));
        DataBytes.Add(Convert.ToByte(Convert.ToChar("o")));
        DataBytes.Add(Convert.ToByte(Convert.ToChar("u")));
        DataBytes.Add(Convert.ToByte(Convert.ToChar("t")));
        DataBytes.Add(Convert.ToByte(0));
string time = Timeout.ToString();
foreach (char ch in time)
            DataBytes.Add(Convert.ToByte(ch));
        DataBytes.Add(Convert.ToByte(0));
    }
    #endregion
     Send (RemoteEndPoint, (byte [])DataBytes.ToArray (typeof
(byte)));
      session = CreateNewSession (RemoteEndPoint, 2,
Filename, "octet", BlockSize);
}
```

限于篇幅 TFTPProcess 类中其他方法不再赘述。

为了在服务器端实现文件传输并发处理，实现了 TFTPManager 对象来管理这些多线程。它以队列 TFTPProcessContainer 的形式对多个报文处理与传输单线程进行统一管理。

下面的源代码实现了 TFTPManager 类的多线程报文处理过

程 (其实际任务执行还是调用了 TFTPProcess 类中的相关方法):

```
privatevoid ProcessDatagram(byte[] ReceivedBytes, int
 NumBytesReceived, IPEndPoint RemoteEndPoint)
{
int CurrOpcode = Convert.ToInt16(ReceivedBytes[1]);
        string EndPointString = ((IPEndPoint)RemoteEndPoint).
Address.ToString() + ":" +
        ((IPEndPoint)RemoteEndPoint).Port.ToString();
if ((CurrOpcode == 1) || (CurrOpcode == 2))
    {
        TFTPProcessContainercontainer= newTFTPProcessCon
tainer();
        container.process = newTFTPProcess(
                FullPath,LevelOfLogger,LevelOfEvent,AllowRRQ,
AllowWRQ,AllowWRQOverwrite,
         AllowOptions,AllowTIDCheck,ResendInterval,Timeout,
TFTPLogger,LocalEndpoint.Address);
              container.threadStart = newThreadStart(container.
process.StartListener);
        container.thread = newThread(container.threadStart);
          container.thread.Name = container.process.GlobalID.
ToString();
        container.thread.IsBackground = true;
        container.thread.Start();
lock (ProcQueueLock)
        {
            ProcessQueue.Add(container);
        }

for (int SleepCounter = 0; SleepCounter <= 5; SleepCounter++)
        {
if (container.process.IsListening)
            {
                                container.process.ProcessDatagram
(ReceivedBytes, NumBytesReceived,

RemoteEndPoint);
this.OnTFTPTransferEvent(newTFTPTransferEventArgs(contai
ner.process));
                container.process.TFTPProcessEvent +=
 newTFTPProcessEventHandler(this.OnTFTPProcessEvent);
                SleepCounter = 10;
            }
else
            {
Thread.Sleep(100);
            }
        }
    }
else
    {
        Send(RemoteEndPoint, Error4);
    }
}
```

为了在 UI 中方便管理 TFTP 服务，TFTPManager 类中定义了两个方法：StartListener () 用于启动 TFTP 服务器监听请求；StopListener () 用于终止 TFTP 服务器监听请求。

下面的源代码实现了 TFTP 服务的管理：

```
publicvoid StartListener()
{
try
    {
        TFTPLogger.Open();
            LocalSocket = newSocket(LocalEndpoint.Address.
AddressFamily, SocketType.Dgram, ProtocolType.Udp);
        LocalSocket.Bind(LocalEndpoint);
        StartStateTimer();
        Loop = true;
        Receive();
    }
catch (SocketException ex)
    {
        Loop = false;
    }
}
publicvoid StopListener()
{
    Loop = false;
if (CheckTimer ! = null)
        CheckTimer.Dispose();
try
    {
if (LocalSocket ! = null)
        {
            LocalSocket.Shutdown  (SocketShutdown.Both); //I
think we want both here unlike the process
            LocalSocket.Close();
        }
    }
catch (ObjectDisposedException ex)
    { }
lock (ProcQueueLock)
    {
for  (int StateIndex = 0; StateIndex < ProcessQueue.Count;
StateIndex++)
        {
            ProcessQueue[StateIndex].process.StopListener();
            ProcessQueue[StateIndex].process.CloseSocket();
        }
        ProcessQueue.Clear();
    }
```

```
    TFTPLogger.Close();
}
```

事件参数有3类：多线程管理事件参数 (TFTPManagerEventArgs)、单线程报文处理事件参数 (TFTPProcessEventArgs)、传输会话管理事件参数 (TFTPTransferEventArgs)，3种参数都通过代理的方法进行参数设置。

TFTPSession 类描述了传输会话过程中的文件 I/O 操作。为了跟踪会话状态，还定义了 TFTPTransferState 类，以保存在连续的 TFTP 传输中的状态信息，它主要用于应用程序中的 UI。

4.2 应用 TFTPServerApp 的实现

TFTPServerApp 是一个简单精致的服务器应用程序。主要用于 TFTP 服务管理以及 TFTP 参数的设置。它启动时，首先加载 TFTP 配置文件，将其反序列化为一个 TFTPParameters 对象，然后根据这些参数启动 TFTP 服务，监听客户端的 TFTP 请求。

下列代码片段实现 TFTP 服务的管理：

```
privatevoid StopTFTPServer()
{
    tftp.StopListener();
if (TFTPServerThread ! = null)
        TFTPServerThread.Abort();
    TFTPServerThread = null;
    timerEvents.Stop();
}
privatevoid StartTFTPServer()
{
try
    {
bool ls = tftp.IsListening;
        TFTPServerThread = newThread (newThreadStart(tftp.
StartListener));
        TFTPServerThread.Name = "TFTP Server Thread";
        TFTPServerThread.IsBackground = true;
        TFTPServerThread.Start();
        timerEvents.Start();
    }
catch (Exception ex)
    {
MessageBox.Show(ex.Message);
    }
}
```

4.3 TFTPClientApp 的实现

TFTPClientApp 是一个简单客户端应用程序，其实现非常简单，核心功能都是调用 TFTPLibrary 中的方法来完成。TFTPClientApp 主要用于测试目的：可在两台机器上同时运行 TFTPClientApp.exe 和 TFTPServerApp.exe，设置好 TFTP 参数(保证一致)，就可以完成文件的上传与下载。笔者是在同一机器上测试，只需要客户端将 TFTP 服务器地址设为本机地址即可。

5 TFTP 在网络管理中的应用

在网络管理中，TFTP 服务器程序常用于网络设备的配置文件 (Configuration) 变更或系统映像 (Image) 升级。大多数应用是基于 DOS 界面的，使用命令启动。在这种应用环境中，网络设备通常充当 TFTP 客户端。

为验证 TFTPServerApp 的有效性，笔者在本机上运行 TFTPServerApp.exe，然后 Telnet 到 TFTP 客户端 (局域网中的一台交换机 DES-3550，地址为 172.20.0.3，Image 文件为 des3550-b37.had)，在其 CLI 界面中从服务器（笔者本机，地址为 202.196.107.2）下载该映像文件对设备成功升级。图 8 显示的是服务器，图 9 显示的是客户端运行结果。

图 8 TFTP 应用：服务器端运行结果

图 9 TFTP 应用：客户端运行结果

（收稿日期：2013-02-21）

基于 ASP.NET 的网上书店设计与实现

李志云

摘　要：网上购物现在越来越普及，网上书店也大量涌现。基于 ASP.NET 技术，给出了网上书店系统的设计与实现。

关键词：网上书店；ASP.NET 编程；C# 语言

1　引言

随着网络的普及，网上书店越来越成为人们购书的重要渠道。通过网上书店可以在网上进行图书展示和销售。文中给出了一个小型网上书店的设计与实现。图 1 是该系统运行时的首页。

图 1　系统运行界面

2　开发环境

Windows7+Visual Studio2008+ SQL Server2005 Express。

3　后台数据库结构

该网站后台数据库采用 SQL Server2005 Express，数据库名称：网上书店。数据库中包括 8 个数据表：图书表、图书类型表、购物车表、订单表、详细订单表、会员表、新闻表和管理员表。图 2-图 9 是这些表的结构。

(1) 图书表：用于保存所有图书信息，其结构如图 2 所示。

图 2　图书表结构

(2) 图书类型表：用于保存图书类型信息，其结构如图 3 所示。

表 - dbo.图书类型表 摘要

列名	数据类型	允许空
[illegible]	char(20)	
类型名	char(20)	✓

图 3　图书类型表结构

(3) 购物车表：用于保存购物车中的图书信息，其结构如图 4 所示。

表 - dbo.购物车表 摘要

列名	数据类型	允许空
[illegible]	int	
会员名	char(12)	✓
图书编号	int	✓
数量	int	✓

图 4　购物车表结构

(4) 订单表：用于保存提交的订单信息，其结构如图 5 所示。

表 - dbo.订单表 摘要

列名	数据类型	允许空
订单编号	int	
会员名	char(12)	✓
订单日期	datetime	✓
发货方式	char(20)	✓
付款方式	char(20)	✓
总金额	float	✓
是否发货	bit	✓
备注	ntext	✓

图 5　订单表结构

(5) 详细订单表：用于保存订单的图书信息，其结构如图 6 所示。

表 - dbo.详细订单表 摘要

列名	数据类型	允许空
[illegible]	int	✓
会员名	char(12)	✓
图书编号	int	✓
数量	int	✓

图 6　详细订单表结构

(6) 会员表：用于保存注册会员的信息，其结构如图 7 所示。

表 - dbo.会员表 摘要

列名	数据类型	允许空
[illegible]	char(12)	
密码	char(32)	✓
姓名	char(20)	✓
性别	char(2)	✓
出生日期	datetime	✓
联系地址	nchar(60)	✓
联系电话	char(13)	✓
手机	char(12)	✓
邮政编码	char(6)	✓
身份证号	char(18)	✓

图 7 会员表结构

(7) 新闻表：用于保存图书新闻信息，其结构如图 8 所示。

表 - dbo.新闻表 摘要

列名	数据类型	允许空
[illegible]	int	
标题	char(60)	✓
内容	ntext	✓
时间	datetime	✓

图 8 新闻表结构

(8) 管理员表：用于后台管理人员的信息，其结构如图 9 所示。

表 - dbo.管理员表 摘要

列名	数据类型	允许空
[illegible]	char(20)	
密码	char(32)	✓
权限	int	✓

图 9 管理员表结构

4 网站的文件夹结构

网站的文件夹结构如图 10 所示。

图 10 网站的文件结构

5 在配置文件中设置到后台数据库的连接

在网站配置文件 web.config 的<appSettings>节中添加如下代码：

```
  <appSettings>
<add key="con" value="server=.\sqlexpress;database=网上书店;integrated security=sspi"/>
</appSettings>
```

6 数据库操作公共类设计

公共类存在于 App_Code 文件夹中，类文件名称：DB.cs。代码如下：

```
public class DB
{
   public SqlConnection Con = new SqlConnection();//创建连接对象
   public SqlCommand Com = new SqlCommand();//创建命令对象
   public SqlDataAdapter Da = new SqlDataAdapter();//创建适配器对象
   public DataSet Ds = new DataSet();//创建数据集对象
   //定义一个用于返回数据库连接字符串的方法
   public String GetConnectionString()
   {
      String ConStr;
       ConStr = System.Configuration.ConfigurationManager.AppSettings.Get(0).ToString();
      return ConStr;
   }
   //定义一个用于返回数据集的公共查询方法
   public DataSet GetDataTableBySql(String SqlStr)
//返回数据集
   {
      Con.ConnectionString = GetConnectionString();
      Com.Connection = Con;
      Com.CommandText = SqlStr;
      Da.SelectCommand = Com;
      try
      {
         Ds.Clear();
         Con.Open();
         Da.Fill(Ds);
         Con.Close();
      }
      catch (SqlException)
      {
         Con.Close();
      }
      return Ds;
   }
```

```
//定义一个用于返回执行数据更新操作是否成功标志的方法
public Boolean UpdateDataBySql(String SqlStr)
{
    Con.ConnectionString = GetConnectionString();
    Com.Connection = Con;
    Com.CommandText =SqlStr;
    try
    {
        Con.Open();
        Com.ExecuteNonQuery();
        Con.Close();
        return true;
    }
    catch (SqlException)
    {
        Con.Close();
        return false;
    }
  }
}
```

7 网站前台页面的母版页设计

网站中的其他页面基本都是依托母版页来实现的。图 11 是前台页面的母版页界面。

图 11 前台页面母版

该母版页采用表格布局。后台 C# 代码如下：

```
String SqlStr;//创建字符串变量
DB db = new DB();//创建类的对象
DataSet Ds = new DataSet();//创建数据集对象
protected void btnLogin_Click(object sender, EventArgs e)
{
        String Md5_User_Pwd = FormsAuthentication.
HashPasswordForStoringInConfigFile (this.txtPassword.Text.
ToString(), "MD5");//作为密码方式加密
    SqlStr = "select * from 会员表 where 会员名='" + this.
txtUsername.Text + "' and 密码='" + Md5_User_Pwd + "'";
    Ds = db.GetDataTableBySql(SqlStr);
    try
    {
    if (Ds.Tables[0].Rows.Count == 0)
    {
        this.Labinfo.Text = "用户名或密码错误,请重试！";
        this.txtUsername.Focus();
    }
    else
    {
        this.Labinfo.Text = " 用户 " + this.txtUsername.
Text + " 恭喜您登录成功！";
        Session["Username"] = this.txtUsername.Text;
//保存用户名到 SESSION 对象中
        }
    }
    catch (Exception)
    {
        this.Labinfo.Text = "没有得到任何数据,请重试！";
    }
  }
  protected void btnRegister_Click(object sender, EventArgs
e)
  {
    Response.Redirect("Register.aspx");
  }
```

8 后台管理登录页面设计

图 12 是后台用户登录时的页面。

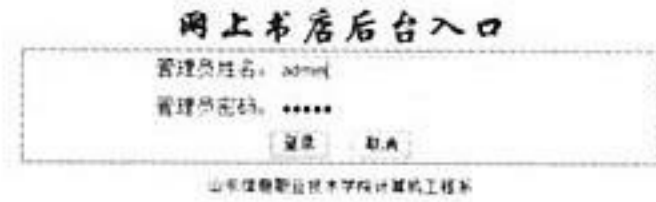


图 12 管理登录页面

其中登录按钮的代码如下：

```
        String Md5_User_Pwd = FormsAuthentication.
HashPasswordForStoringInConfigFile (this.txtPassword.Text.
ToString(), "MD5");//作为密码方式加密
    SqlStr = "select * from 管理员表 where 用户名='" + this.
txtUsername.Text + "' and 密码='" + Md5_User_Pwd + "'";
    ds = db.GetDataTableBySql(SqlStr);
    try
    {
    if (ds.Tables[0].Rows.Count == 0)
    {
        Response.Write("<script>alert('用户名或密码错
误,请重试！')</script>");
        this.txtUsername.Focus();
    }
    else
    {
        Session ["AdminUsername"] = this.txtUsername.
Text;
```

```
            Response.Write ("<script>window.location.href=
'Default.aspx';</script>");
        }
    }
    catch (Exception)
    {
        Response.Write("<script>alert('没有得到任何数据，
请重试！')</script>");
    }
```

9 后台管理页面的设计

后台管理页面依然采用母版页设计，图 13 是后台管理中的图书新增页面。

图 13 管理界面

其中“新增图书”按钮的代码如下：

```
protected void btnAdd_Click(object sender, EventArgs e)
  {
    if (Session["AdminUsername"] ! = null)
    {
        try
        {
            string path_file = fulPicture.PostedFile.FileName.
ToString();
            string file_type = path_file.Substring (path_file.
LastIndexOf("."));
            string file_name = DateTime.Now.Year.ToString()
+ DateTime.Now.Month.ToString () + DateTime.Now.Day.
ToString () + DateTime.Now.Hour.ToString () + DateTime.
Now.Minute.ToString() + DateTime.Now.Second.ToString();
            full_name = file_name + file_type;
            string path = Server.MapPath (" ~/image/") +
full_name;
            fulPicture.SaveAs(path);
        }
        catch (Exception)
        {
            Response.Write("<script>alert('上传文件失败！')<
/script>");
        }
        string typeID = ddlBookType.SelectedValue.ToString
();//获得图书类型编号
        string image_path = "~/image/" + full_name;
        SqlStr = "insert into 图书表 (类型编号,图书名,价格,
作者,开本,印张,字数,版次,书号,印数,图片)"
            + "values ('" + typeID + "','" + txtBookName.Text
+ "','" + txtPrice.Text + "','"
            + "'" + txtWriter.Text + "','" + txtSize.Text + "','"
            + "'" + txtYZ.Text + "','" + txtWords.Text + "','"
            + "'" + txtBC.Text + "','" + txtISBN.Text + "','"
            + "'" + txtNum.Text + "','" + image_path + "')";
        try
        {
            if (db.UpdateDataBySql(SqlStr))
            {
                Response.Write("<script>alert('图书新增成功！
')</script>");
            }
            else
            {
                Response.Write("<script>alert('图书新增失败！
')</script>");
            }
        }
        catch (Exception)
        {
            Response.Write("<script>alert('图书新增失败')</
script>");
        }
    }
    else
    {
        Response.Redirect("..\\error.aspx");
    }
}
```

10 结语

在网上书店的开发中，许多页面使用了母版页，这样可以使页面风格统一，而且提高了制作效率。对后台数据库操作时通过公共类来实现，简化了代码的编写。

参考文献

[1] 宁云智，刘志成，等. ASP.NET 程序设计实例教程. 人民邮电出版社，2011.

（收稿日期：2013-02-22）

利用 PGF&Tikz 实现 PDF 教学演示动画

刘 烽

摘 要：介绍了一种利用 LATEX 的 PGF&Tikz 绘画宏包制作 PDF 幻灯片演示动画的一种方法，实现简单，动画质量高，特别适合于教学演示的动画制作。

关键词：LATEX 系统；Beamer 宏包；PGF&Tikz 宏包；演示动画

1 引言

从长期的教学实践中会发现，如果在教学中适当加入某些动画，对促进理解，加深印象具有很大的帮助。然而，目前许多“可见即可得”的动画制作软件（Flash，PPT）等在精确作图方面仍有所欠缺，如对数学公式支持不够好，动画的播放与控制操作较难等。近期，笔者在学习 LATEX 排版系统时发现，利用 Beamer 和 PGF&Tikz 等宏包可以制作出表现力相当强的动画，特别适合于日常的教学演示动画。

2 LATEX 简介

LATEX 是一种基于 Tex 的文档排版系统，由美国计算机专家 Leslie Lamport 开发。利用 LATEX 可以在短时间内生成高质量的文档，特别是针对复杂数学公式的科技类文档，LATEX 较之其他的排版软件更有优势。LATEX 自从开发以来，经过多年的发展，从最初的 LATEX2.09 版本发展到如今的 LATEX2e，其性能越来越完善，功能越来越强大，已经深受排版从业人员特别是科技文稿排版人员的亲睐。

图 1 CTEX 编辑工具 WinEdt 程序界面

和其他许众多软件一样，LATEX 在不同的硬件和操作系统上有不同的实现版本。Unix/Linux 下的系统主要是 teTEX，而 Windows 下则有 MiKTEX 和 fpTEX。而在国内，人们常用支持中文的 Tex 系统——CTEX。CTEX 是中文套装的简称，是把 MiKTEX 和一些常用的工具，如 Gsview，WinEdt 等工具打包在一起制作的一个简易安装程序，并对其中的中文支持部分进行了配置，使得安装后马上就可以对中文文档进行排版。最新版本的 CTEX 可以从 CTAN 论坛（http://www.ctex.org）下载获得。图 1 是 CTEX 的主要编辑工具 WinEdt 的程序界面。

3 制作 PDF 幻灯片的 Beamer 宏包和 PGF&Tikz 绘画宏包

LATEX 宏包是由许多基本命令组成的指令集，利用宏包可以大大扩展 LATEX 的功能。Beamer 就是 LATEX 下制作 PDF 幻灯片的一个著名宏包，它的主要特点是：（1）可以直接使用 pdflatex 编译，也可以使用 dvips 编译，不需要其他后处理程序。（2）同标准 LATEX 兼容良好。（3）可以方便创建许多精美动画及漂亮的演示效果。（4）可以轻松将幻灯片转化为书籍等其他格式的文档。以下代码来自于 beamer 使用手册[1]的一个简单例子：

```
\documentclass{beamer}
% This is the file main.tex
\usetheme{Berlin}
\title {Example Presentation Created with the Beamer Package}
\author{Till Tantau}
\date{\today}
\begin{document}
\begin{frame}
    \titlepage
\end{frame}
\section*{Outline}
    \begin{frame}
    \tableofcontents
\end{frame}
\section{Introduction}
    \subsection{Overview of the Beamer Class}
    \subsection{Overview of Similar Classes}
```

```
\section{Usage}
    \begin{frame}
     \end {frame} % to enforce entries in the table of
contents
\end{document}
```

PGF&TikZ 是在 LATEX 系统中的画图宏包之一，由于 PGF&TikZ 和 Beamer 的作者都是出自 Till TanTau 教授之手，所以 Beamer 和 PGF&TikZ 结合使用非常的完美。利用 PGF&TikZ 可以制作出色彩锐丽，清晰美观的图形，特别是对精确图（如数学函数）的绘制，简单方便，出图快捷。另一方面，由于 PGF&TikZ 的易用性，人们在此基础上开发了许多特殊功能的宏包，如制作动画的宏包 animate、书写特殊公式的宏包 siunitx 等，使得 PGF&TikZ 通过几个简单的命令就可以绘制出具有生动动画的图形。表 1 是 PGF&TikZ 中几个常用基础命令[2]：

表 1　PGF&TikZ 中的常用命令

命令	含义	实例	实例说明
\draw	绘画命令	\draw [->] (0,0) -- (1,0);	绘制一个向右的箭头
circle	绘制圆形	\draw [color=red] (0, 0)circle (.5) ;	以（0，0）为圆心绘制半径为 0.5 个单位的圆形
rectangle	绘制长方形	\draw (0, 0) rectangle (2, 1);	绘制一个长 2 个单位，宽为 1 个单位的长方形
ellipse	绘制椭圆	\draw (0, 0) ellipse (.7 and 0.5) ;	以（0，0）为中心绘制长、短轴为 0.7、0，5 的椭圆形
node	绘制节点	\path[draw](0, 0) node {A} -- (1,0) -- (1,1) node {B};	绘制两个节点，并进行标注
\shade	绘制阴影或渐变色	\shade [top color=red, bottom color=green] (0,0) rectangle (2,1) ;	绘制一个长 2 个单位，宽为 1 个单位的长方形，内部为红色到绿色的垂直渐变填充色

4　PGF&TikZ 编写数学函数演示动画

本例以动画的形式演示高等数学中积分和导数的几何意义，其源代码如下所示：

```
\documentclass[cjk]{beamer} %使 Beamer 幻灯片支持中文
  \mode<presentation>
  {
   %定义 Beamer 幻灯片的主题
    \usetheme{Warsaw}
  }
  \usepackage{CJK} %中文支持宏包
  \usepackage{ifthen} %逻辑判断宏包
  \usepackage{tikz} %PGF&TikZ 绘图宏包
  \usetikzlibrary{shapes,arrows}
  \hypersetup{pdfpagemode=FullScreen} %使编译生成的
%PDF 文件进行预览
  \usepackage{animate} %产生动画效果宏包
  \usepackage{siunitx} %数学公式样式宏包
\begin{document}
 %设置变量
 \newcounter{r}
 %自定义命令
 \newcommand{\escalar}[1]
 {
      \setcounter{r}{#1 * #1 * #1* #1}
 }
 \newcounter{m}
 \setcounter{m}{0}
  \newcounter{mc}
  \newcounter{ang}

  \begin{CJK*}{GBK}{kai}
  \CJKtilde
  \begin{frame}
  \frametitle{函数导数与积分的几何意义演示动画}
      \begin{center}
          \begin {animateinline} [loop, poster = first,
controls, palindrome]{2}
              \whiledo{\them < 21}
              {
                  \begin {tikzpicture} [scale =0.55,
domain=-1.5:2]
                      %自定义 pgf 宏命令，用于计
%算函数 y=x^3 的导数与积分等运算
                  \pgfmathsetmacro{\x}{0.1*\them}
                \pgfmathsetmacro{\y}{0.001*\them*\them*\them}
                  \pgfmathsetmacro{\rx}{2*cos(atan(3*\x*\x))}
                  \pgfmathsetmacro{\ry}{2*sin(atan(3*\x*\x))}
                    \pgfmathsetmacro{\ra}{atan(3*\x*\x)}
                      %绘制 y=x^3 数学函数曲线
                  \draw[red,thick,<->] plot (\x,{\x*\x*\x})
                    node[below right] {$y=x^3$};
                      %绘制网格
                  \draw[loosely dotted] (-3,-4) grid (3,8);
                      %x 轴
                      \draw [-> ] (-4.25,0) --
(4.25,0) node[right] {$x$};
                      %y 轴
                      \draw [-> ] (0, -4.25) --
(0,8.25) node[above] {$y$};
                      %绘制 x,y 轴的刻度线
                  \foreach \x/\xtext in {1/1, 2/2, 3/3}
                      \draw [shift={(\x,0)}] (0pt,2pt)
-- (0pt,-2pt) node[below] {\tiny$\xtext$};
                      \foreach \x/\xtext in {-1/-1, -
2/-2, -3/-3}
                      \draw [shift={(\x,0)}] (0pt,2pt)
-- (0pt,-2pt) node[below] {\tiny$\xtext$};
                      \foreach \y/\ytext in {1/1, 2/2,
3/3, 4/4,5/5,6/6,7/7,8/8}
```

```
                \draw  [shift={(0,\y)}] (2pt,0pt)
-- (-2pt,0pt) node[left] {\tiny$\ytext$};
                \foreach \y/\ytext in {-1/-1, -
2/-2, -3/-3, -4/-4}
                \draw  [shift={(0,\y)}] (2pt,0pt)
-- (-2pt,0pt) node[left] {\tiny$\ytext$};
                %求积分时用到的运算
                \setcounter   {mc} {\value {m}
*\value{m}*\value{m}}
                %求导数时用到的运算
                \setcounter  {ang}{3*\value{m}
*\value{m}}
                 %绘制一个封闭图形,表示积
%分运算的几何意义
                \shade[top color=blue,bottom
color=yellow!  50]
                        (0,0) parabola bend (0,0)
(0.1*\them,0.001*\themc) |- (0,0);
                \escalar{\them}
                %显示积分运算公式
            \draw [blue](5.5cm,2pt) node[above]

{$\displaystyle\int\limits_0^{\them/10} \!  \!  x^3\mathrm{d}x =
                        \displaystyle\frac
{\ther}{4000}$};
                %绘制函数 y=x^3 上一个点
                \shade   [ball   color  =green]
(0.1*\them,0.001*\themc) circle (2pt);
                %动画开始帧判断
                \ifthenelse{\them > 0}
                {
                        \draw  [blue,thick,->,>=
latex](\x-\rx,\y-\ry)--(\x+\rx,\y+\ry);
                        \draw  [blue,dashed] (\x,
\y)--(3+\x,\y);
                         \draw  [->, >=stealth']
(\x+0.6,\y)arc(0:atan(3*\x*\x):0.6);
                }{}
            %动画从第 9 帧开始显示角度值
                \ifthenelse{\them > 9}
                {
                        \draw   [blue](\x+0.6,\y+
0.6) node{$\alpha$};
                        \draw    [blue]  (5.5cm,
5.5cm) node[above]{ $\alpha =\ang{\ra}$};
                }{}
        \end{tikzpicture}
        \stepcounter{m}
        \ifthenelse{\them < 21}
        {
            \newframe
        }{}
            } % 结束 While
        \end{animateinline}
    \end{center}
  \end{frame}
  \end{CJK*}
\end{document}
```

将上述源代码拷贝至 WinEdt 中，点击菜单 Accessories-->PDF-->pdfLaTex 编译源文件，编译成功后即可生成 PDF 文件，通过 Adobe Reader 阅读器即可播放此动画。从最终生成的 PDF 幻灯片看（如图 2），其操作界面类似于普通播放器，设置有播放、快进、快退、暂停按钮，并可随意控制播放速度，整个图形色彩鲜艳，线条细腻，美观大方。不难发现，此动画很好地阐述了积分和导数几何意义的概念。

函数导数与积分的几何意义演示动画

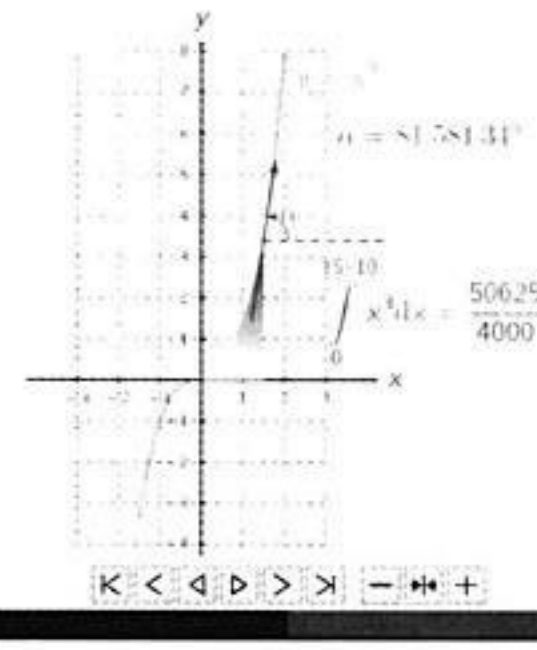


图 2 程序运行结果

5 结语

利用 PGF&TikZ 制作 PDF 演示动画，方便快捷，图形精美，可直接生成 PDF 文件，且演示时带有类似播放器的功能，方便动画的控制。但是，这种方法的缺点也比较明显：即并非“所见即所得”的制作动画方式，需要动画设计人员具有一定的编程功底。即便如此，通过 PGF&TikZ 制作动画也不失为一种较好的方法。

参考文献

[1] The beamer class Manual for version 3.07.

[2] PGF&TikZ Graphics for LATEX.

[3] LATEX2e 及常用宏包使用指南.

（收稿日期：2012-12-12）

基于 OpenGL 的三维坐标系运动轨迹捕捉

张兴沛

摘　要：基于 OpenGL 技术，利用 VC++开发模拟三维坐标系，并利用串口通信的方式接收数据，在三维坐标系统进行绘制数据轨迹线。

关键词：OpenGL 技术；三维坐标系；坐标系旋转；串口通信；VC++语言

1　引言

目前，随着城市化的快速发展，需地下施工作业的工程越来越多，由于地下作业清晰定位作业的方位比较困难，容易偏离既定方位，因此，在计算机模拟的三维空间中绘制地下施工作业的轨迹对地下施工作业十分重要。本系统主要用于解决地下作业方向定位问题，可实时将地下作业机器的运行轨迹在模拟的三维坐标系中绘制出来，真正解决地下作业的方向定位问题。

2　功能模块

本系统主要由三维空间轨迹捕捉模块、三维坐标的投影模块、串口通信模块 3 个部分构成，对于捕捉到的轨迹数据能保存为二进制文件，并能讲保存的数据文件重新导入系统。同时对于三维坐标系可以使用鼠标 (键盘的上下左右键) 进行旋转，以满足从不同的角度来观察设备运行的轨迹，为了能够更加细致的关系运行数据，使用鼠标的转轮可以放大或缩小模拟三维坐标系的大小，基本满足地下施工需要，如图 1 所示。

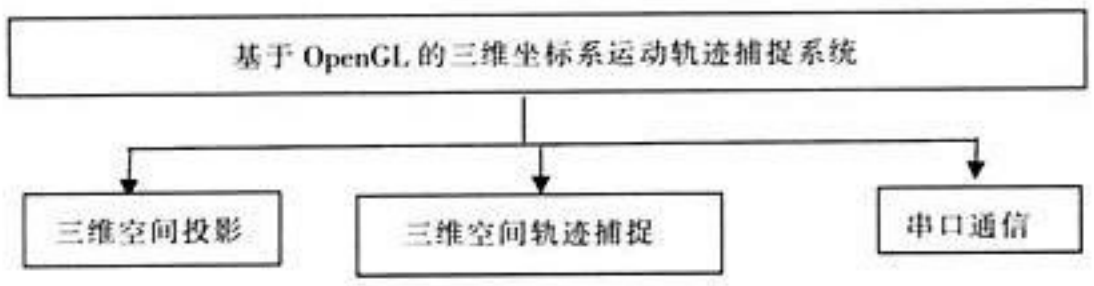


图 1　系统功能模块

3　编程思想

3.1　OpenGL 的初始化

OpenGL 是一个开放的三维图形软件包，它独立于窗口系统和操作系统，所以使用 OpenGL 开发的应用程序可以在各种平台间移植；主要代码如下：

```
Void  InitOpenGL(Void)
{
    //置黑背景
    glClearColor(0.0,0.0,0.0,0.0);
            glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFF
ER_BIT);
    glShadeModel(GL_SMOOTH);
    // 设置混色函数取得半透明效果
    glBlendFunc(GL_SRC_ALPHA,GL_ONE);
    glEnable(GL_BLEND);
    //平滑线条
  glEnable (GL_LINE_SMOOTH);
  glEnable (GL_BLEND);
    //充许深度测试
    glEnable(GL_DEPTH_TEST);
    glDepthFunc(GL_LEQUAL);
    //清除颜色缓冲和深度缓冲
      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFF
ER_BIT);
    //用单位矩阵替换当前矩阵
    //复位
    glLoadIdentity();
    //旋转角度
    glRotatef(xrof,1.0f,0.0f,0.0f);//x 轴
    glRotatef(yrof,0.0f,1.0f,0.0f);//y 轴
    glRotatef(zrof,0.0f,0.0f,1.0f);//z 轴
}
```

3.2　利用 OpenGL 中的 glVertex3f () 函数模拟绘制三维空间中的三维坐标系

主要代码如下：

```
void DrawCooridate(float m_Zoom)
{
    glLineWidth(2.0f*m_Zoom);
glBegin(GL_LINES);
//X 轴
glColor3f(1.0f,1.0f,1.0f);
glVertex3f(-11.0f*m_Zoom,0.0f,0.0f);
glVertex3f(11.0f*m_Zoom,0.0f,0.0f);
//Y 轴
glColor3f(1.0f,1.0f,1.0f);
glVertex3f(0.0f,-11.0f*m_Zoom,0.0f);
glVertex3f(0.0f,11.0f*m_Zoom,0.0f);
```

```
    //Z 轴
    glColor3f(1.0f,1.0f,1.0f);
    glVertex3f(0.0f,0.0f,-11.0f*m_Zoom);
    glVertex3f(0.0f,0.0f,11.0f*m_Zoom);
    glEnd();
}
```

3.3 绘制三维坐标系中数据在 3 个面的投影坐标系

主要代码如下：

```
void DrawCoordinate (CDC* pDC, int width, int height, int multiple)
{
    pDC->SetTextAlign(TA_LEFT);
    //绘制横轴
    pDC->MoveTo(0,height/2.0);
    pDC->LineTo(width,height/2.0);
    //绘制纵轴
    pDC->MoveTo(width/2.0,3.0);
    pDC->LineTo(width/2.0,height-3.0);
    //绘制横轴箭头
    pDC->MoveTo(width,height/2.0);
    pDC->LineTo(width-5.0,height/2.0-5.0);
    pDC->MoveTo(width,height/2.0);
    pDC->LineTo(width-5.0,height/2.0+5.0);
    //绘制纵轴箭头
    pDC->MoveTo(width/2.0,3.0);
    pDC->LineTo(width/2.0-5.0,8.0);
    pDC->MoveTo(width/2.0,3.0);
    pDC->LineTo(width/2.0+5.0,8.0);
    //标注坐标原点
    pDC->TextOutA(width/2.0+10.0,height/2.0+10.0,"0");
}
```

3.4 在坐标系中绘制读取的数据轨迹线

```
//绘制模拟三维坐标系中的数据轨迹线
void DrawDataLine(CMy3DtrackDoc* pDoc, int multiple, float m_Zoom)
{
    CGrap* m_Grap=new CGrap;
   int isFirst=1;//用于判断是否为从数据链表的第一数据取值
    float x1,y1,z1,x2,y2,z2;//用于划线时的坐标传递
    float stepDis,angel1,angel2,angel3;
    glPushMatrix();
    if(! pDoc->m_list.IsEmpty())
    {
        glLineWidth(2.0*m_Zoom);
        glColor3f(0.0f,1.0f,0.0f);
        glBegin(GL_LINES);
        POSITION pos=pDoc->m_list.GetHeadPosition();
        for(int j=0;j<pDoc->m_list.GetCount();j++)
        {
            if(isFirst==1)
            {
                x1=0;
                y1=0;
                z1=0;
                isFirst=0;
            }
            else
            {
                x1=x2;
                y1=y2;
                z1=z2;
            }
            m_Grap =(CGrap*)pDoc ->m_list.GetNext (pos);
            stepDis=m_Grap->stepDis;
            angel1=m_Grap->m_Xangle;
            angel2=m_Grap->m_Yangle;
            angel3=m_Grap->m_Zangle;
            z2=stepDis*sin(angel1);
            x2=stepDis*sin(angel2);
            y2=stepDis*sin(angel3);
            //把坐标转换成三维坐标系中的坐标
            z2=z2*10/(200*multiple);
            x2=x2*10/(200*multiple);
            y2=y2*10/(200*multiple);
            x2=x2+x1;
            y2=y2+y1;
            z2=z2+z1;
                glVertex3f (x1*m_Zoom,y1*m_Zoom,z1*m_Zoom);
                glVertex3f (x2*m_Zoom,y2*m_Zoom,z2*m_Zoom);
            }
            glEnd();
        }
        glPopMatrix();
}
```

绘制投影坐标系中的数据轨迹线：

```
void DrawDataLine (CDC* pDC,CMy3DtrackDoc* pDoc,float width,float height,int &multiple,CString w)
{
    pDC->SetMapMode(MM_ANISOTROPIC);
    pDC->SetWindowOrg(-(200.0*width*multiple)/(width-20.0),(200.0*height*multiple)/(height-20.0));
        pDC ->SetWindowExt ((2*200.0*width*multiple)/(width-20.0),-(2*200.0*height*multiple)/(height-20.0));
    pDC->SetViewportOrg(0,0);
    pDC->SetViewportExt(width,height);
    CGrap* m_Grap=new CGrap;
      int isFirst=1;//用于判断是否为从数据链表的第一数据
//取值
    float x1,y1,z1,x2,y2,z2,tempx,tempy,tempz;//用于划线时
```

```
//的坐标传递
    float stepDis,angel1,angel2,angel3;
    if(! pDoc->m_list.IsEmpty())
    {
        POSITION pos=pDoc->m_list.GetHeadPosition();
        for(int j=0;j<pDoc->m_list.GetCount();j++)
        {
            if(isFirst==1)
            {
                x1=0;
                y1=0;
                z1=0;
                isFirst=0;
        }
        else
        {
                x1=x2;
                y1=y2;
                z1=z2;
        }
m_Grap=(CGrap*)pDoc->m_list.GetNext(pos);
        stepDis=m_Grap->stepDis;
        angel1=m_Grap->m_Xangle;
        angel2=m_Grap->m_Yangle;
        angel3=m_Grap->m_Zangle;
        z2=stepDis*sin(angel1);
        x2=stepDis*sin(angel2);
        y2=stepDis*sin(angel3);
        tempx=x1+x2;
        tempy=y1+y2;
        tempz=z1+z2;
        if (abs (tempx) >=200*multiple ||abs (tempy) >=
200*multiple||abs(tempz)>=200*multiple)
        {
            multiple=multiple+1;
            pDoc->UpdateAllViews(NULL);
        }
        if(w=="X")//如果是在 XY 面的投影
        {
            x2=x1+x2;
            y2=y1+y2;
            pDC->MoveTo(x1,y1);
            pDC->LineTo(x2,y2);
        }
    }
    else if(w=="Y")//如果是在 YZ 面的投影
    {
        y2=y1+y2;
        z2=z1+z2;
        pDC->MoveTo(y1,z1);
        pDC->LineTo(y2,z2);
    }
    else if(w=="Z")//如果是在 ZX 面的投影
    {
        x2=x1+x2;
        z2=z1+z2;
        pDC->MoveTo(x1,z1);
        pDC->LineTo(x2,z2);
    }
        }
    }
}
```

3.5 串口接收数据

主要代码如下：

```
LRESULT ReceiveData (WPARAM wParam, LPARAM
lParam)
{
    DWORD res;
    HANDLE hCom;
    DWORD dwError = 0;
    unsigned char temp;
    int n=0;
  bool IsRead;
    COMSTAT rst;
    ClearCommError(hCom,&res,&rst);
    memset(&rOverLaped,0,sizeof(OVERLAPPED));
    rOverLaped.hEvent =CreateEvent (NULL,TRUE,FALSE,
NULL);
    n=rst.cbInQue;
    DWORD S=n;
    CString str="";
    CString receiveData="";
    for(int i=0;i<n;i++)
    {
                IsRead =ReadFile (hCom,&temp,1,&res,
&rOverLaped);
        if(IsRead)
        {
            str.Format("%02X ",temp);
            receiveData+=str;
        }
        else
        {
            switch (dwError = GetLastError())
                {
                    case ERROR_IO_PENDING:
                    {
                        WaitForSingleObject
(rOverLaped.hEvent,2000);
                        IsStop=false;
                        break;
                    }
                default:
                    {
```

```
                                break;
                            }
                        }
                    }
                return true;
}
```

3.6 程序输出

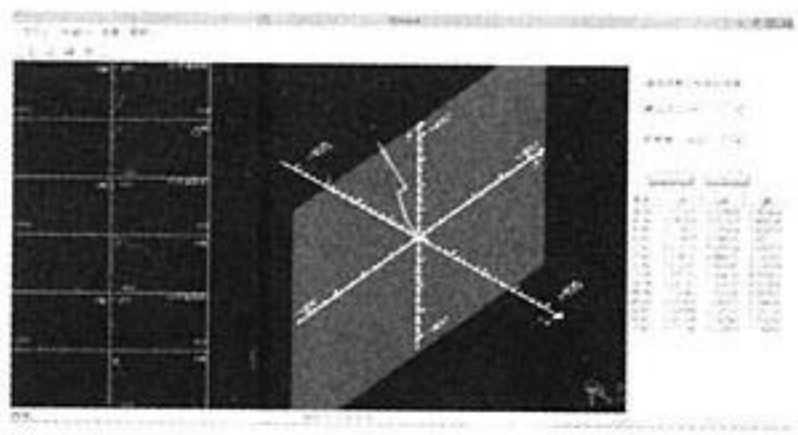


图 2 程序输出结果

结果如图 2 所示（图 2 中展示的数据是一次实际测量的数据）：

4 结语

利用 VC++开发平台与 OpenGL 结合起来开发模拟三维空间的运动轨迹捕捉只需要简单的几个步骤，即可模拟出地下作业的设备实际运行轨迹，有利于解决地下作业定位难得问题，从而彻底解决地下作业方向偏差问题，同时 OpenGL 是一个与硬件无关的软件接口，开发独立于窗口系统和操作系统，以它为基础开发的应用程序可以十分方便地在各种平台间移植，为在不同的操作系统平台运行提供了方便，期望这套系统能为可视化作业的各类工程项目提供一个完美的解决方案。

（收稿日期：2012-12-12）

（上接第 57 页）

```
                    <tr>
<%
                Else ' 别人发的消息
%>
                    <tr>
                        <td bgcolor = " <% =lineColor% > "
><% =strSender% > 对所有人说：<% If strBiaoQing <> "wu"
Then %><img src = "biaoqing/<% =strBiaoQing% > .gif" ><%
End If %><%=strMsg%></td>
                    <tr>
<%
                End If
%>
<%
            End If
            ' 清空数据
            strSender=""
            strBiaoQing=""
            strMsg=""
        End If
Next
%>
</table>
</body>
</html>
```

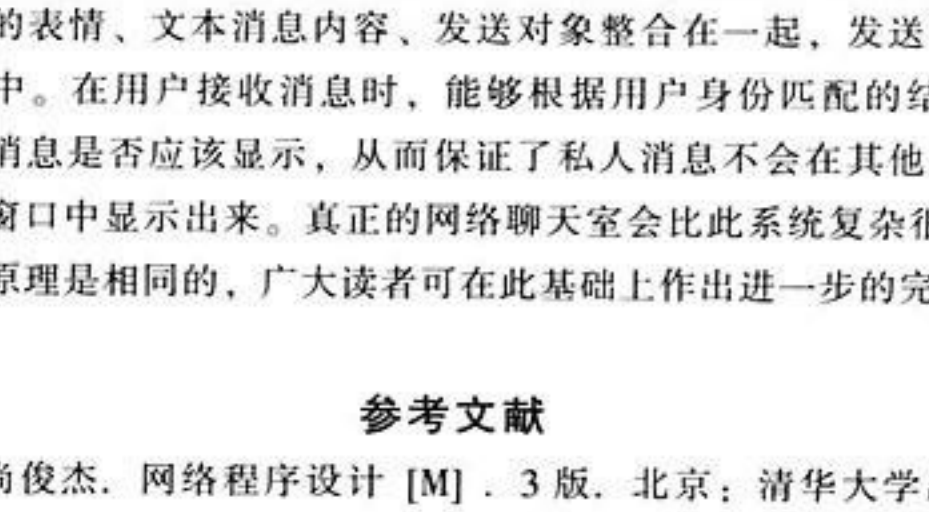


图 6 用户的个人聊天室

5 系统测试

通过登录 3 个用户“10010”、“10011”、“10012”进行测试，分别发送了公共消息、私人消息，消息的发送与接收均正常。其中“10010”用户的聊天室测试主界面如图 6 所示。

6 结语

使用 Application 对象，记录了聊天室中的聊天内容，并在页面中刷新和显示。对聊天内容进行特定的封装，能够将用户发送的表情、文本消息内容、发送对象整合在一起，发送到服务器中。在用户接收消息时，能够根据用户身份匹配的结果，判定消息是否应该显示，从而保证了私人消息不会在其他人的聊天窗口中显示出来。真正的网络聊天室会比此系统复杂很多，但其原理是相同的，广大读者可在此基础上作出进一步的完善。

参考文献

[1] 尚俊杰. 网络程序设计 [M] . 3 版. 北京：清华大学出版社，2008.

[2] 管西京. ASP+SQL Server 动态网站开发案例开发学习笔记 [M] . 北京：电子工业出版社， 2008.

[3] 郭常圳. ASP 网络开发例学与实践 [M] . 北京：清华大学出版社，2005.

（收稿日期：2012-12-08）

Java 版井字棋的设计与实现

仇 宾

摘　要：井字棋是大家所熟知的一个小游戏，虽然简单，但其中包含了一些编程的基本技巧和基本算法，在 Eclipse 环境下用 Java 语言编写一个可以人人、人机对弈的井字棋游戏。

关键词：Java 语言；井字棋；游戏编程

1　引言

井字棋，即棋盘是一个井字，是一种在 3X3 格子上进行的连珠游戏，和五子棋比较类似，由于棋盘一般不画边框，格线排成井字而得名。游戏规则很简单，游戏双方一方为“X”，一方为“O”，哪方率先实现 3 子相连即为胜者。如图 1 所示。

图 1　正在进行中的井字棋游戏

现在来对井字棋游戏的代码实现做一个探讨，首先介绍人人对弈方式的实现过程，然后在此基础上介绍人机对弈井字棋的实现。

2　人人对弈

2.1　难点释疑

人人对弈实现起来较为简单，游戏双方交替在棋盘上落下棋子“X”或“O”即可，最大问题就是如何判定胜负。从棋盘可以看出，获胜（即任一方出现三连子）一共有 8 种情况：3 连横、3 连竖以及 2 个斜对角，如果我们给每个落子点从 0 到 8 编号，如图 2 所示。

那么，这 8 种获胜情况我们可以用一个二维数组来表示：

```
static final int[][] WIN_STATUS = {
        {0, 1, 2},
        {3, 4, 5},
        {6, 7, 8},
        {0, 3, 6},
        {1, 4, 7},
        {2, 5, 8},
        {0, 4, 8},
        {2, 4, 6}
};
```

图 2　棋盘落子点编号

这样，再定义一个一维数组，每走一步棋就对上面的二维数组进行遍历——从二维数组中依次取出 8 种情况放入一维数组，然后查看一维数组中的 3 个棋子是否相同，如果相同可以判定获胜。

2.2　设计与实现

第一步：写一个类继承自 JFrame，然后定义几个必要的变量和常量，如下：

```
public class Tic extends JFrame {
  JButton[] jb = new JButton[9];// 按钮数组构成棋盘的 8
//个落子点
  static final char empty = 32; // 代表空格
  static int clicknum = 0;      // 记录单击次数,决定是走 X
//还是走 O
  static final int INFINITY = 100;// 带标无穷值
  static final int WIN = +INFINITY;     // O 获胜
  static final int LOSE = -INFINITY;    // X 获胜
```

```
static final int DRAW = 0; // 平局
// 获胜棋盘状态
static final int[][] WIN_STATUS = {
        {0, 1, 2},
        {3, 4, 5},
        {6, 7, 8},
        {0, 3, 6},
        {1, 4, 7},
        {2, 5, 8},
        {0, 4, 8},
        {2, 4, 6}
    };
}
```

第二步：构建棋盘。

在 Tic 类的构造方法中，JFrame 的布局方式设置为 GridLayout，然后每个格子里放置一个按钮即可，代码如下：

```
public Tic(){
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(400, 400);      //棋盘大小
        this.setLayout(new GridLayout(3,3));
        this.setTitle("井字棋");
        // 让窗口居中显示
        Dimension screen = Toolkit.getDefaultToolkit().
    getScreenSize();    // 获取屏幕尺寸封装到 screen 中
    this.setLocation((screen.height - this.getHeight())/2,
        (screen.width - this.getWidth())/2); // 窗口居中
        // 把按钮加入窗体
        for(int i=0; i<9; i++){
            jb[i] = new JButton("");
            jb[i].setFont(new Font("Arial",Font.BOLD,30));
        jb[i].addActionListener(new JBClick());// 事件监听
            this.add(jb[i]);
        }
        this.setVisible(true);
    }
```

第三步：经过前两步，现在加上 main 方法，就可以执行显示棋盘了，代码如下：

```
public static void main(String[] args) {
        new Tic();
JOptionPane.showMessageDialog (null,"O 代表玩家 1,X 代表玩家 2,玩家 1 先走","提示",JOptionPane.DEFAULT_OPTION);
    }
```

但是按钮还没有事件响应，现在单击按钮没有任何反应。下面给按钮添加事件响应。每次单击按钮需要完成如下任务：因为是两个玩家进行游戏，所以按钮要轮流显示“X”和“O”，表示游戏双方交替走棋。每次单击走棋后还要判定是否有获胜方，有则提示哪方获胜，游戏结束，无则继续走棋。代码如下：

```
private class JBClick implements ActionListener{
        // 当单击按钮时
public void actionPerformed(ActionEvent e) {
        for(int i=0; i<9; i++){
if(e.getSource() == jb[i]){// 哪个按钮被单击
        if(++clicknum % 2 == 0){//偶数次是 X
                    jb[i].setText("X");
                }else{
                    jb[i].setText("O");//奇数次是 O
                }
jb[i].setEnabled(false);//被单击过的按钮不可用
                }
            }
    int gamestate = gameState(jb);// 调方法获取棋盘状态
            // 输出棋局胜负
            switch (gamestate) {
            case WIN:
JOptionPane.showMessageDialog(null, "O 方获胜", "提示",
JOptionPane.DEFAULT_OPTION);
                break;
            case LOSE:
JOptionPane.showMessageDialog(null, "X 方获胜", "提示",
JOptionPane.DEFAULT_OPTION);
                break;
            case DRAW:
JOptionPane.showMessageDialog (null, "平局", "提示",
JOptionPane.DEFAULT_OPTION);
                break;
            }
            // 如果结束,则提示
if (gamestate == WIN || gamestate == LOSE || gamestate
== DRAW) {
                int over = JOptionPane
        .showConfirmDialog(null, "是否再来一局?", "提示",

JOptionPane.YES_NO_OPTION,JOptionPane.
QUESTION_MESSAGE);
                if (over == JOptionPane.YES_OPTION)
{
                    for (int i = 0; i < 9; i++) {
                        jb[i].setText(" ");
                        jb[i].setEnabled(true);
                    }
                } else {
                    System.exit(0); // 退出游戏
                }
            }
        }
    }
```

其中代码：int gamestate = gameState (jb) ;表示调用 gameState 方法来判定是否有获胜方。如何判定呢？首先，遍历整个棋盘，看棋盘是否已经满，如果未满，就直接返回，继续走棋，如果满了，则判断是否符合 8 种数组中的一种，符合则分出胜负，不符合则说明平局。代码如下：

```
public int  gameState(JButton[] jb){
        int result = 1; // 棋局状态初始值
        boolean isFull = true;        // 棋盘是否已满
        // 判断棋盘是否已满
        for (int pos = 0; pos < 9; pos++) {
              char chess = jb[pos].getText().charAt(0);
              if (empty == chess) {
                    isFull = false;
                    return result; // 未满则返回,继续走棋
              }
        }
        // 棋局已满,判定胜负
for(int[] status:WIN_STATUS){//遍历 8 中棋局获胜状态
              //得到某个获胜棋局状态的第一个索引的字符
              char chess = jb[status[0]].getText().charAt(0);
// 如果为空,说明此处未下棋子,跳出循环,找下一个状态
              if(chess==empty){
                    continue;
              }
              int i ;
              //不为空,则看其余两子是否与其相同
              for(i=1; i<status.length; i++){
            if(jb[status[i]].getText().charAt(0)!  =chess){
              break;      //不同则跳出循环
                }
              }
              if(i==status.length){  // 三子连线
                    result = chess== 'O' ? WIN : LOSE;
                    break;
              }
        }
        if (result !  = WIN & result !  = LOSE & isFull)
        result = DRAW;       //不输不赢且棋盘满则为平
        return result;
}
```

调用该方法后，如果返回值 result 等于 1，说明棋局没有下满，继续走棋；等于 WIN，说明走“O”的一方获胜；等于 LOSE，说明“X”方获胜；等于 DRAW，则平局。

执行过程如图 3 所示。

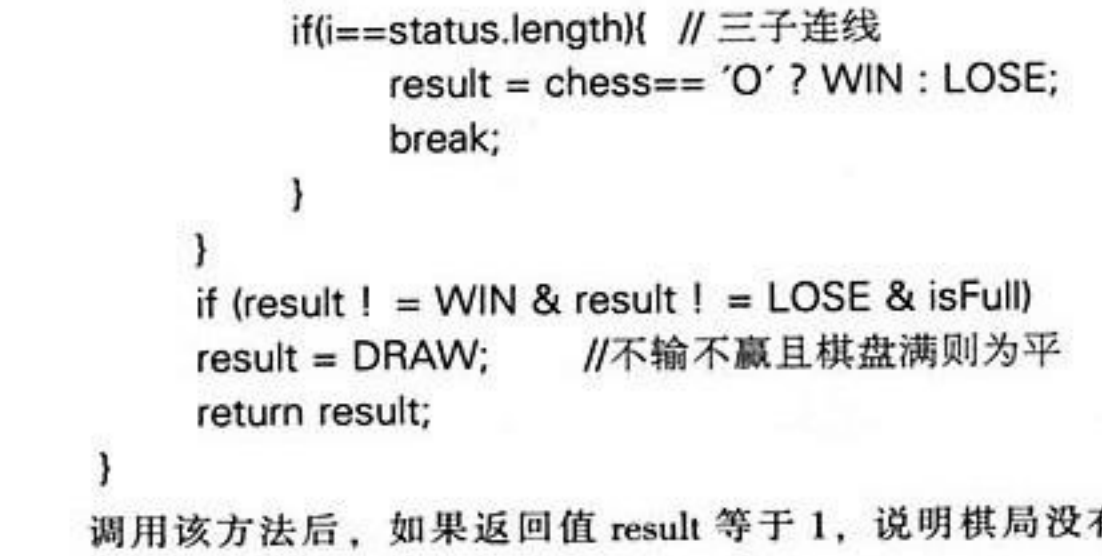


图 3　O 方获胜

2.3　编程技巧

尽管很简单的一个小程序，但是其中却包含一些编程技巧，比如使用一个二维数组来记录棋局获胜的情况，如果不用数组，则需要使用 8 个 if 语句来描述 8 种获胜情况。再比如程序中用到的 for each 语句，很轻松地做到了用一个一维数组去遍历二维数组。如果使用 for 语句，那么这个循环将会很难写。

3　人机对弈

3.1　难点释疑

人机对弈的难点在于当人走一步棋之后，计算机如何走下一步，即计算机如何找出最合适的位置去走棋。这就需要一定的算法，或者叫做计算机的 AI。对于井字棋、五子棋等两方较量的游戏来说，Minimax 算法（极小极大算法）是最基本也是最常用的。算法的原理不在这里解释了，直接看该算法在井字棋中的应用。

井字棋中，假设使用“X”的是人，使用“O”的是计算机。“X”方先走，设定 X 方的最大利益为正无穷（程序使用常量+INFINITY 表示），O 方的最大利益为负无穷（程序中使用-INFINITY 表示），即 X 方和 O 方走的每步棋都要力图使自己的利益最大化，而使对方的利益最小化。这样称 X 方为 MAX（因为它总是追求更大的值），O 方为 MIN（它总是追求更小的值），各自都为争取自己的最大获益而努力。现在举例说明，比如图 4 所示的棋局树。

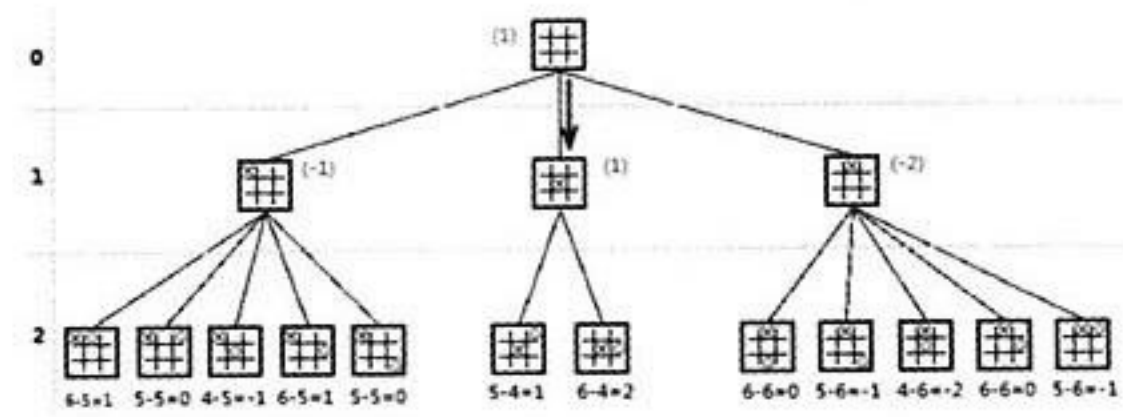


图 4　棋局形成的树

X 方先走，有 3 种选择，如图 4 中第二层所示。假设 X 方选择最左边的走法，那么 O 方接下来将有 5 种走法，O 方会选择最小化的走法，即值为-1 的走法，因为它的最大利益是负无穷；同理，X 方的另外两种走法会分别得到 O 方的最小值 1 和-2。这样，对于 X 方来说，3 种走法会导致 O 方最小化值分别为-1、1、-2，X 方的最佳策略则是选择其中最大的，即第二层中间的走法，因为它的最大利益是正无穷，这就是极小极大算法的体现——X 方的选择总是极大化，O 方的选择总是极小化。

对于其中那些值是如何计算的举例说明，比如对于第 3 层最左边的棋局，在这种状态下，如果把棋局空白处都填上 X，则 X 共有 6 中 3 连子情况，即获胜情况；如果把空白处都填上

O，则 O 共有 5 种 3 连子情况，所以结果是二者相减等于 1。

在具体走棋过程中，MAX 面对 MIN 最大获利中的最小值时，会选择其中最大的，比如图 4 第二层小括号内的值都是第三层中能使 MIN 最大获利的最小值，这时候 MAX 选择其中最大的，这对 MAX 最为有利，所以 MAX 方选择图 4 第二层中间的走法最好。同样道理，MIN 也会一样，选择对自己最有利的，即 MAX 有可能获得最大值。这时候，MIN 在走棋时会考虑 MAX 方占据哪个位置对 MAX 最有利，然后 MIN 把这个位置先占了。有点难理解，其实就是抢先把对对手有利的位置抢占了。

简单说，X 方或者 MAX 方的走棋是由人来控制的。对于 O 方或者 MIN 方，它走棋时要考虑哪个位置对 X 方最有利，然后把该位置占据，即 O 的最佳走棋就是 X 的最佳走棋。所以 O 在走棋之前，先站在 X 的角度寻找最佳走棋位置。后文中 minimax 方法就是站在 X 角度来考虑极小极大算法，找到 X 的最佳走棋位置，然后由 O 方来占据该位置。

3.2 极小极大算法

整个算法包括如下几个部分：

首先要有一个评估方法 gameState，对每走一步棋后的棋局进行评估，估值为 WIN 常量说明 X 方，即 MAX 方获胜；估值为 LOSE 则 O 方，即 MIN 方获胜；估值 DRAW 为平局；估值为 INPROGRESS，说明棋未走完；估值为 DOUBLE_LINK，说明棋局中有两连子情况。

然后用一个 minimax 方法寻找在当前棋局状态下 X 方的最佳位置，X 方的最佳位置就是当 X 走该位置后，O 方所有走法中最小值里的最大值，比如图 4 中第二层 X 的位置选择。当找到该位置后，由 O 方来抢先占据该位置。

最后用两个递归方法 MIN 和 MAX 来遍历所有的棋局。MIN 方法负责找出 O 方的最小值，比如图 4 第二层最左边的棋局会导致 5 中 O 方的走法，MIN 方法就是找出这 5 种走法中的最小值。同理，MAX 方法负责找出 X 方的最大值，比如图 4 第二层 3 种棋局中的中间棋局。

3.3 代码实现

在刚才那个人人对弈井字棋的基础上，进行一些修改来实现人机对弈。首先单击事件加入调用方法让计算机走棋的代码，如下：

```
// 按钮的监听事件
private class JBClick implements ActionListener {
    // 当单击按钮时
    public void actionPerformed(ActionEvent e) {
        for (int i = 0; i < 9; i++) {
            if (e.getSource() == jb[i]) {
        jb[i].setText("X"); // 被单击的按钮走"X"
                jb[i].setEnabled(false); //置为不可用
                }
            int gamestate = gameState(jb); // 获取棋盘状态
                // 如果棋局未结束,则计算机走下一步
if (! (gamestate == WIN || gamestate == LOSE || gamestate
== DRAW)) {
                int nextpos = getNextMove(jb); // 获取下一步
//走棋位置
                jb[nextpos].setText("O"); // 走棋"O"
                jb[nextpos].setEnabled(false);
    gamestate = gameState(jb); // 获取最新的棋盘状态
                }
                // 输出棋局胜负
                switch (gamestate) {
                case WIN:
 JOptionPane.showMessageDialog(null, "X 方获胜", "提示",
JOptionPane.DEFAULT_OPTION);
                  break;
                case LOSE:
                    JOptionPane.showMessageDialog (null,
"O 方获胜", "提示",JOptionPane.DEFAULT_OPTION);
                    break;
                case DRAW:
      JOptionPane.showMessageDialog(null, "平局", "提示",
JOptionPane.DEFAULT_OPTION);
                    break;
                }

                // 如果结束,则提示
    if (gamestate == WIN || gamestate == LOSE ||
gamestate == DRAW) {
                int over = JOptionPane
    .showConfirmDialog(null, "是否再来一局？", "提示",
     JOptionPane.YES_NO_OPTION,
        JOptionPane.QUESTION_MESSAGE);
    if (over == JOptionPane.YES_OPTION) {// 再来一局
                for (int i = 0; i < 9; i++) {
                    jb[i].setText(" ");
                    jb[i].setEnabled(true);
                }
            } else {
                System.exit(0); // 退出游戏
                }
                }
        }
}
```

然后，获取棋局状态的方法加入寻找两连子的代码，如下：

```
// 获取棋盘当前状态
public int gameState(JButton[] jb) {
    int result = INPROGRESS;
    boolean isFull = true;
```

```
        // 判断棋盘是否已满
        for (int pos = 0; pos < 9; pos++) {
            char chess = jb[pos].getText().charAt(0);
            if (empty == chess) {
                isFull = false;
            }
        }
        // 寻找三连子情况
for (int[] status : WIN_STATUS) {// 遍历 8 中棋局获胜状态
        // 得到某个获胜棋局状态的第一个索引的字符
            char chess = jb[status[0]].getText().charAt(0);
      // 如果为空,说明此处未下棋子,跳出循环,找下一个状态
            if (chess == empty) {
                continue;
            }
            int i;
        for (i = 1; i < status.length; i++) {// 查看其余两个字符
if (jb[status[i]].getText().charAt(0) ! = chess) { // 不与第一个索
//引字符一致
                    break; // 表明未三子连线,跳出
                }
            }
            if (i == status.length) { // 三子连线
                result = chess == 'X' ? WIN : LOSE;
                break;
        }
}
// 寻找两连子情况
if (result ! = WIN & result ! = LOSE) {

        if (isFull) {
        result = DRAW;       //不输不赢且棋盘满则为平
        } else {
        int[] finds = new int[2];// 存放 X 或 O 的两连子情况
            for (int[] status : WIN_STATUS) {
                char chess = empty;
                boolean hasEmpty = false;
                int count = 0;   // 计数
                for (int i = 0; i < status.length; i++) {
            if (jb[status[i]].getText().charAt(0) == empty) {
                hasEmpty = true;     // 该处没有棋子
            } else {
            if (chess == empty) {// 有棋子
                chess = jb[status[i]].getText().charAt(0);
            }
            if (jb[status[i]].getText().charAt(0) == chess) {
                count++;  //且棋子相同则加 1
            }
        }
}
if (hasEmpty && count > 1) {
        if (chess == 'X') {
                finds[0]++;
            } else {
                finds[1]++;
            }
        }
}
// 两连子情况
if (finds[1] > 0) {// O 的两连子
        result = -DOUBLE_LINK;
} else if (finds[0] > 0) {      // X 的两连子
        result = DOUBLE_LINK;
    }
}
}
return result;    // 记录了胜负平或者两连子情况
}
```

O 方走棋时，要得到走棋位置，用一个方法来获取该位置，如下：

```
public int getNextMove(JButton[] board) {
    int nextPos = minimax(board, 3);
    return nextPos;
}
```

上面方法中调用了极小极大算法 minimax，如下：

```
//以 'X' 的角度来考虑的极小极大算法
public int minimax(JButton[] board, int depth) {
    int[] bestMoves = new int[9];//存放最佳走棋位置
    int index = 0;
    int bestValue = -INFINITY;
    // 搜索所有空位,试探填上 X,然后选其中最小值的
    for (int pos = 0; pos < 9; pos++) {
        if (board[pos].getText().charAt(0) == empty) {
            board[pos].setText("X");
            int value = min(board, depth);// 得到最小值
            if (value > bestValue) {// 选择最小值里最大的
                bestValue = value;
                index = 0;
                bestMoves[index] = pos;
            } else if (value == bestValue) {
                index++;
                bestMoves[index] = pos;
            }
            board[pos].setText(" ");
        }
    }
return bestMoves[index];
}
```

最后，两个递归方法 min 和 max 如下：

```
//对于 'O',估值越小对其越有利
public int min(JButton[] board, int depth) {
    int evalValue = gameState(board);
boolean isGameOver =  (evalValue == WIN || evalValue ==
```

```
LOSE || evalValue == DRAW);
        if (depth == 0 || isGameOver) {
            return evalValue;
        }
        int bestValue = INFINITY;
        for (int pos = 0; pos < 9; pos++) {
            if (board[pos].getText().charAt(0) == empty) {
                board[pos].setText("O");
                // 选择最小值
            bestValue = Math.min (bestValue, max(board,
depth – 1));
                board[pos].setText(" ");
            }
    }
    return evalValue;
}
//对于 'X',估值越大对其越有利
public int max(JButton[] board, int depth) {
    int evalValue = gameState(board);
    boolean isGameOver =  (evalValue == WIN || evalValue
== LOSE || evalValue == DRAW);
        if (depth == 0 || isGameOver) {
            return evalValue;
        }
        int bestValue = –INFINITY;
        for (int pos = 0; pos < 9; pos++) {
            if (board[pos].getText().charAt(0) == empty) {
                board[pos].setText("X");
                // 选择最大值
            bestValue = Math.max (bestValue, min (board,
depth – 1))board[pos].setText(" ");
            }
    }
    return evalValue;
}
```

4 结语

人人对弈较为简单，把对胜负判定的代码写好就行了，人机对弈需要考虑算法，让计算机来计算下一步走棋的位置，这比较复杂一些，尽管用了很大的篇幅来讲解极小极大算法，可能还是不太好理解，需要大家对照代码进一步仔细思考了。

还有一点在文中没有交代，就是搜索棋局过程中，限定了搜索的深度，即在 min 和max 方法中通过 depth 变量来控制的，程序中限定 depth 是 3，即搜索 3 层，因为每增加一层，棋局状态都会成几何指数的增长，层数太多会加大计算机的计算时间。这里不再详细探讨了，有兴趣的可以查看相关资料。

这里介绍的算法是最基本的，还有在此基础上的剪枝算法、负极小极大算法等，大家可以深入地去进一步学习。

（收稿日期：2013-02-18）

网游巨头云聚分享传统游戏转型之路

一份由艾瑞咨询提供，2012 年第一季度中国手机网游市场规模突破 10 亿元，同比增长超过五成，手机网游用户规模超过 1 亿，同比增长 34.8%。另外 CNNIC 也显示，2012 年我国手机网民数量为 4.2 亿，年增长率达 18.1%，远超网民整体增幅。此外，网民中使用手机上网的比例也继续提升，由 69.3%上升至 74.5%。

2012 年传统网络的发展依旧光鲜亮丽，但是增速下降以及总量增加比不高说明了目前传统网络发展已经从爆发期向缓和期过渡。在整体网民增速放缓的时代，更多的电脑用户在已经使用手机上网来取代原有上网的模式。依附于传统互联网的网游面临增量放缓的大环境，如何寻求突破成为 2013 年的关键点。

面对手机游戏这块还未完全开发的蛋糕，以及不断增长的用户付费数都吸引不同厂商加入这一领域。作为传统互联网“吸金大户”的网游巨头同样不想错过，如何利用自己原有的资源，搭上移动互联网的快车成为他们考虑的重要问题。面对着 2013 中国移动广告将达 16 亿美元的诱惑，金山软件副总裁兼金山游戏总裁西山居游戏 CEO 邹涛、昆仑万维创始人兼 CEO 周亚辉、光宇游戏总裁宋洋三家传统网游巨头云聚 GMGC 第二届全球移动游戏大会，并围绕“传统厂商的转型之路”主题与众多移动互联网业内人士，及开发者、运营商、终端厂商等各家关键合作伙伴共同畅谈讨论。

第二届 GMGC 全球移动游戏大会将于 5 月 6 日在北京国家会议中心召开，大会由全球移动游戏联盟 GMGC 主办，全面邀请来自全球移动游戏产业生态链的相关者参与，会场将全球移动游戏开发者训练营品及国际推广与合作专场、创业世界杯–全球移动游戏应用评选大赛等多个板块，同时全球移动游戏联盟将通过组织 GMGC 全球移动游戏大会、沙龙、比赛等多种形式的交流和推广活动，把游戏玩家、开发商、运营商、平台商等齐聚一堂，共商合作大计，为移动游戏产业共建和谐健康的生态链！

端口转发技术实现局域网穿透（上）

杨 勇

摘 要：局域网内的主机，可以经网关由网络地址转换 (NAT) 访问公网，而公网主机要访问内网，必须要在网关上作端口映射，或者安装反向代理服务器。根据端口反弹原理，构建一个能穿越 NAT 的转发系统，先由内网主动发起对公网主机的 TCP 连接，建立数据传输通道，经由这条数据通道，将外网客户端与内网服务器的通信数据进行转发，转发系统由两个转发器构成，分别用 select 模型和 IOCP 管理多个 TCP 套接字上的数据传输，实现了公网主机无需利用网关的协助便可访问内网主机。

关键词：端口转发；TCP 协议；NAT 技术；select 模型

1 引言

由于 IP 地址数量有限，同时也为了数据安全，学校、大中型企业经常将集团内部电脑连成局域网。局域网内电脑分配私有 IP 地址，局域网再通过防火墙和网关连接到外界广域网。局域网内电脑要访问外网，必须经过网关上的 NAT 进行网络地址转换，但反过来，外网电脑只能连接到局域网的网关，而无法穿过网关直接连到内网主机，因此，在内网主机上建立的各种网络服务，例如，Web 服务器、FTP 服务器、SQLserver 数据库服务等等，只有同一个局域网内的主机可以连接，而外网主机无法访问。当然，这正是数据安全的需要。但是某些特定情况下，人们也需要在外网上获取内网的服务。目前，有几种技术可从外网访问内网主机：

(1) 静态端口映射 (Static Port Mapping)，在 NAT 网关上开放一个固定端口，然后设定此端口收到的数据转发到内网的某个 IP 和端口，并保持映射关系始终存在。外网访问网关上的固定端口，即可获取内网某 IP 上的网络服务。

(2) 反向代理 (Reverse Proxy)，在网关上运行反向代理服务器，接收外网客户端的连接请求，然后将请求转发给内网上的服务器；并把服务器的应答返回给外网客户端。

(3) VPN (Virtual Private Network)，在网关上建立 VPN 服务器，可以在外网客户机和内网之间建立一条虚拟的数据专用通道，外网客户机被虚拟成内网中的一个节点，其实质是 VPN 服务器进行外网主机和内网主机之间通信数据的转发。

上述几种方法的缺点是，均要在网关上设置或安装服务器，许多情况下，非网关管理员无法操作网关，或出于安全需要不能改动网关配置。因此，需要找到一种无需借助网关的辅助，而实现外网访问内网电脑的方法。

为此，设计了一个转发系统，利用端口反弹原理与端口数据转发，以局域网内的主机作为中转，不用在网关上作任何额外的设置，实现从外网访问内网服务的功能。

2 内网端口转发系统框架

局域网的特点是外网不能连接局域网主机，而内网连接外网主机一般不受防火墙阻拦，否则，局域网将无法访问外网。根据这个原理可以采取“端口反弹”原理，即先由内网中的一台主机主动发起对外网主机的 TCP 连接，一旦 TCP 连接建立，则这个连接的两端在接受数据和发送数据上处于完全对等的地位，即双方都可以向对方发送数据，也可以接受对方的数据。这样内网和外网之间建立起一个数据双向传输通道，借用这个通道，把外网客户对内网服务器的访问请求数据，内网服务器对请求的应答数据进行中转，实现外网对内网服务的间接访问。根据该思路，整个端口转发系统框架如图 1 所示。

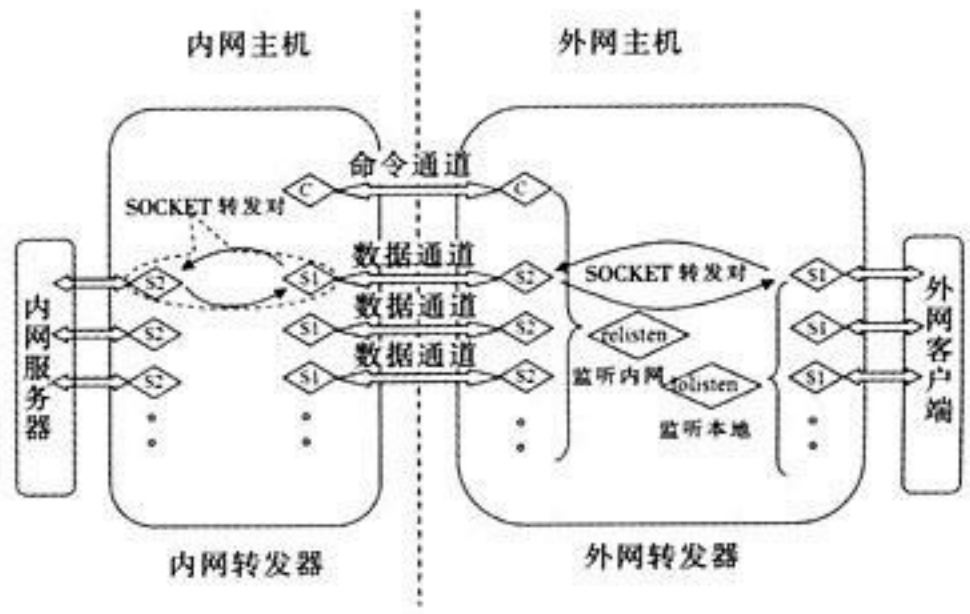


图 1 端口转发系统框架

转发系统由两个转发器组成，现以具体实例说明其工作原理：

（1）内网转发器，在内网中的主机上运行。首先，查找外网地址为 58.59.103.23:6000 的主机，一旦外网主机 58.59.103.23 接受连接请求，则建立套接字 C，并把第一个套接字 C 作为双方的控制命令传输通道，接着继续与外网主机连续建立多个连接套接字 S1，形成多条数据传输通道，供外网转发器在之后的数据传输中使用。当某条数据通道 S1 上传来外网的请求数据，内网转发器开始发起对内网 Web 服务器10.10.1.30:80 的连接请求，建立套接字 S2，S1 和 S2 构成构成一个 Socket 转发对，转发数据。不同的连接对应不同的转发对，数据转发完毕，两个连接关闭，Socket 转发对也随之撤销。

（2）外网转发器，位于具有公网 IP 地址的电脑上，它的工作是，首先，同时建立两个监听 Socket，relisten 和 lolisten，relisten 监听来自内网转发器的连接请求，监听端口设定为 6000。lolisten 则监听来自于本地客户端 Web 浏览器（与外网转发器在同一个电脑上）的请求，监听端口 7000。relisten 首先等待内网请求，内网连接请求首次到达后，建立套接字 C，把第一个套接字 C 作为双方的控制命令传输通道。接着，内网转发器会与外网转发器连续建立多个套接字 S2，外网转发器将建立好的套接字存入连接池（Connecting Pool）以备用。准备就绪后，再向本地客户发出通知，可以开始转发服务了。loelisten 接到本地客户请求后，建立套接字 S1，再从连接池取出一个连接内网的套接字 S2，与之构成 Socket 转发对，把数据转发到内网，再由内网转发器转发到内网服务器上。经过内外两个转发器共 4 个套接字的两次转发，外网请求传到内网服务器上，应答数据也可由这条通道回传到外网客户端上。为了协调两个转发器的工作，第一个连接套接字 C，作为双方互传控制命令的通道单独使用。

3 外网转发器的设计

3.1 并发请求 I/O 调度模型的选择

客户端的请求往往是并发请求，典型的如 Web 服务。对客户每一个请求要建立一个 Socket 转发对，当数据转发完毕，Socket 转发对也随之关闭。因此转发系统必须对大量的连接套接字进行高效率地管理。Windows 系统对并发连接的管理主要有以下几种 I/O 模型：

（1）阻塞套接字+多线程模型。一个套接字对应一个线程处理数据 I/O，但过多的线程使系统资源开销过大，I/O 效率下降。

（2）非阻塞套接字+单线程 select 多路复用模型。select 函数可以对多个套接字进行轮询，返回一个有 I/O 事件的套接字集合，然后对集合中的套接字进行处理，处理完毕后，再进行下一轮轮询。整个轮询和数据处理过程放入单个线程同步执行。

（3）完成端口（IOCP）+重叠 I/O 模型。Windows 系统下效率最高的处理模型。可以并发处理海量的连接请求。

外网转发器只面对一个客户，连接数不多，采用 select 模型。内网转发器则要处理多客户的多个连接，采用完成端口模型构建。外网转发器工作流程如图 2 所示。

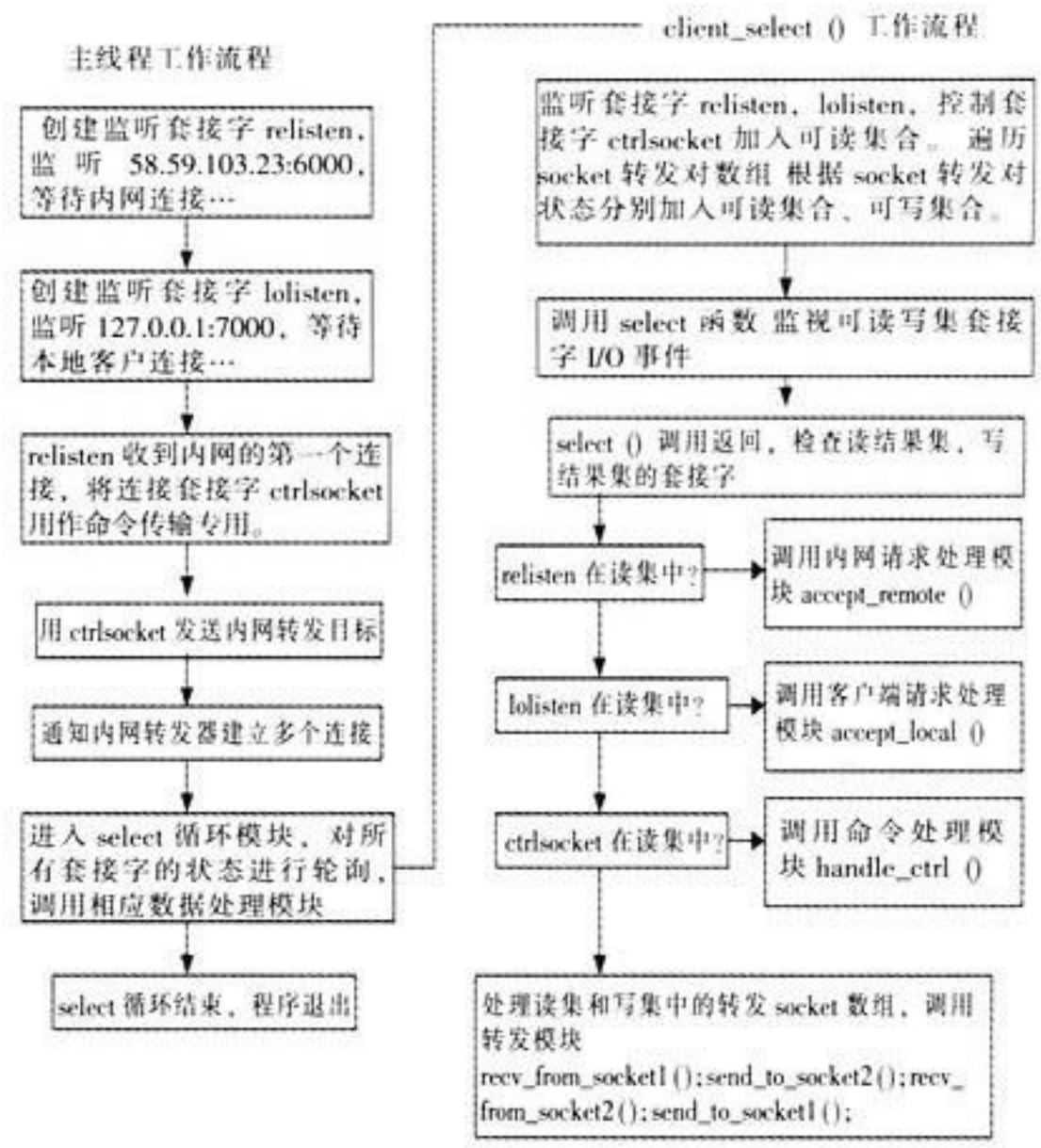


图 2 外网转发器工作流程

3.2 外网转发器主线程

外网转发器主线程代码如下，实现图 2 所示工作流程：

```
void  ClientStart(DataModule* pM)
{
    if (! ClientInit(pM) ) return;
    unsigned int id;  //发送邮件模块
     HANDLE  h  =(HANDLE) (  _beginthreadex (NULL,0,
&SendEmailThread,(LPVOID)pM ,0,&id) );
    CloseHandle(h);
      pM->relisten=create_socket();
    pM->lolisten=create_socket();
    if( !  create_listener(pM->relisten,pM->tranport,true) )
{
        pM->sendmsg(" 端口 %s 监听失败,退出运行.");
        return;
    }
    if( !  create_listener(pM->lolisten,pM->clientport,false)
)   {
        pM->sendmsg(" 端口 %s 监听失败,退出运行.");
        return;
    }
    //等待远程第一个连接 socket ,作为通信 socket
    pM->sendmsg( " 开始监听 端口: %d,%d\n", pM->
tranport,pM->clientport );
```

```
    struct sockaddr_in client;
    int size=sizeof(struct sockaddr);
    if ( (pM->ctrlsocket = accept (pM->relisten, (struct
sockaddr*)&client, &size)) == -1)
        printf( " accept control socket failed! \n");
    else {
  SetSocketNonBlocked( pM->ctrlsocket); //设为非阻塞型
      SetSocketKeepAlive(pM->ctrlsocket,60); //每 60 秒发
//心跳包
      pM->create_loaddr();  //发送转发目标
      send( pM->ctrlsocket,(char*)(&pM->loaddr),16,0);
      ::Sleep(100);
      pM->ctrlbuf[0]='+';
      send( pM->ctrlsocket,pM->ctrlbuf,16,0); //通知内网
//建立多个连接,用于数据转发
      pM->sendmsg( "与内网建立连接! \n");
    }
    while(pM->runstate == pM->eRUN)  //select 循环调用
    {
        pM->client_select();
    }
}
```

当主线程与内网连接，首先建立套接字 ctrlsocket 以传输调度信息，将准备转发的内网服务器 IP 和端口填入 struct sockaddr_in 套接字结构体 pM->loaddr，在 ctrlsocket 上发送到内网。因此，转发目标是由客户端来确定的，根据客户的需要可以灵活地修改。接着由 ctrlsocket 传递命令给内网转发器：预先建立多个连接，以供后续数据转发使用。

3.3 Socket 转发对

Socket 转发对及套接字状态标志分别由几个数组构成，关联数据以同一个数组索引联系起来，各个数组大小均为 128 个元素，各数组元素的功能说明如下：

SOCKET sockets1 [i]：连接本地客户的套接字，用于接收客户请求数据和发送内网应答数据；

SOCKET sockets2 [i]：连接内网转发器的套接字，用于发送客户请求数据和接收内网应答数据；

sockets1 [i] 和 sockets2 [i]：构成一对转发对，用以转发数据。

char* sendbuf [i]：请求数据缓冲区首地址指针，存放请求数据。

char* recvbuf [i]：应答数据缓冲区首地址指针，存放应答数据。

s1recvbyte [i]：套接字 sockets1 [i] 已收到的请求数据字节数。

s2sentbyte [i]：套接字 sockets2 [i] 已发送完毕的请求数据字节数。

s2recvbyte [i]：套接字 sockets2 [i] 已收到的应答数据字节数。

s1sentbyte [i]：套接字 sockets1 [i] 已发送完毕的应答数据字节数。

allclosed [i]：全关闭标志，sockets1 [i] 和 sockets2 [i] 均已关闭。

s1closed [i]：sockets1 [i] 关闭标志

s2closed [i]：sockets2 [i] 关闭标志

closing [i]：半关闭标志，sockets1 [i] 和 sockets2 [i] 中有一个套接字已关闭。

unused [i]：半连接标志，sockets2 [i] 与内网连接已建立而 sockets1 [i] 与本地连接尚未建立，转发对处于待用状态。

各数据和转发函数对应关系如图 3 所示。

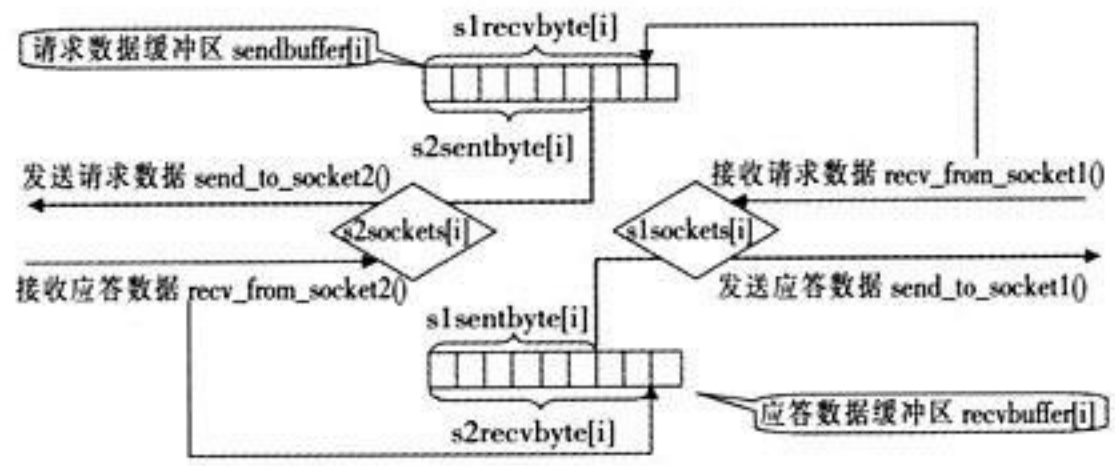


图 3 SOCKET 转发对

3.4 Select 循环模块

Select 模块是整个外网转发器的调度中心，由该模块给各个数据处理函数分派任务，实现图 2 所示的工作流程。其代码如下：

```
void  DataModule::client_select()
{
    int i;
    fd_set readfds, writefds;
    FD_ZERO(&readfds);
    FD_ZERO(&writefds);
    FD_SET(lolisten, &readfds);
    FD_SET(relisten, &readfds);
    if(runstate == eRUN)
            FD_SET(ctrlsocket, &readfds);
      for (i = 0; (i < SOCK_SIZE); i++)
    {
       if (allclosed[i])       continue;
       if (closing[i]) {
      if (! s1closed[i])  FD_SET(sockets1[i], &writefds);
       if (! s2closed[i]) FD_SET(sockets2[i], &writefds);
       }
       if ((! s1closed[i]) && (s1recvbyte[i] < BUFSIZE ))
//请求数据缓冲区未满
                  FD_SET(s1sockets[i], &readfds);
         if ((! s2closed[i]) && (s2sentbyte[i] < s1recvbyte[i]))
//请求数据未发送完
```

```
                FD_SET(s2socekts[i], &writefds);
        if ((! s2closed[i]) && (s2recvbyte[i] < BUFSIZE ))
                FD_SET(sockets2[i], &readfds);
        if ((! s1closed[i]) && (s1sentbyte[i] < s2recvbyte[i]))
                FD_SET(sockets1[i], &writefds);
        }
    select(0, &readfds, &writefds, NULL, NULL);
//select 多路复用
        if (FD_ISSET(relisten, &readfds))  accept_remote();
    if (FD_ISSET(lolisten, &readfds))  accept_local();
    if (FD_ISSET(ctrlsocket, &readfds))  handle_ctrl();
    for (i = 0; (i < SOCK_SIZE); i++)
    {
            if (allclosed[i])   continue;
      if(! s2closed[i]) {
 if (FD_ISSET(sockets2[i], &readfds))recv_from_socket2(i);
      }
      if(! s1closed[i]) {
 if (FD_ISSET(sockets1[i], &writefds)) send_to_socket1(i);
      }
            if (! s1closed[i]) {
 if (FD_ISSET(sockets1[i], &readfds))  recv_from_socket1(i);
        }
      if (! s2closed[i]) {
                if (FD_ISSET(sockets2[i], &writefds))
send_to_socket2(i);
        }
          if (s1closed[i] && s2closed[i])
                        allclosed[i] = true;
      }
}
```

在调用 select 函数之前，遍历检查套接字状态标志数组，根据 Socket 对的状态选择性的加入可读句柄集或者可写句柄集。分以下几种情况：（1）Socket 对全关闭状态，allclosed [i] =true，不用读也不用写；（2）Socket 对半关闭，closing [i] = true，其中未关闭的 Socket 不用再接收数据了，因为另一个 Socket 已经关闭，不能再发送数据。但是未关闭的 Socket 还可以继续发送数据（如果数据还未发送完），所以未关闭的 Socket 加入写句柄集。（3）Socket 对全连接状态，如果数据缓冲区未满，则 Socket 可以继续接收数据，将之加入可读集 readfds。如果已接收的数据未发送完，则 Socket 还需要继续发送数据，将之加入可写集 writefds。

select 函数返回后，循环遍历整个 Socket 转发对数组，如果 Socket 在读集 readfds 中，则表明该 Socket 有数据到达，需要接收。调用与之对应的接收函数，如果 Socket 在写集 writefds 中，则表明该 Socket 可以发送数据，调用与之对应的发送函数。

select 函数及调用的数据处理函数都在主线程中运行，读写函数同步执行，避免了数据同步问题。

3.5 内网请求处理模块 accept_remote ()

accept_remote () 用于从处理监听套接字 relisten 上来自内网的连接请求：

```
void DataModule::accept_remote()
{
    struct sockaddr_in client;
    int size=sizeof(struct sockaddr);
    SOCKET nfd = accept(relisten, (struct sockaddr*)&client,
&size);
    if (nfd == INVALID_SOCKET)  return;
     ioctlsocket(nfd, FIONBIO, &j);  //设连接 SOCKET 为非阻
//塞型套接字
    SetSocketKeepAlive(nfd,60); //每 60 秒发心跳包
    int index =-1;
    for (int j = 0; j < SOCK_SIZE; j++) { //查找 SOCKET 转发
//对数组空闲位置
                if (allclosed[j]) {
                    index = j;
                    break;
                }
    }
    if (index == -1)   {
        closesocket(nfd);
        return;
    }
    sockets2[index] = nfd;   //与内网的连接套接字 nfd 放入
//数组空闲位置
    init_iopos(index);   //关联数据初始化
    allclosed[index] = false;
    closing[index]   = false;
    s1closed[index]  = true;  //s1socket 尚未连接,设置为 true
    s2closed[index]  = false;
    unused[index]    = true;  //index 位置的套接字转发对处于
//待用状态
        connectsum++;  //与内网连接总数递增
}
```

本系统所有套接字均设定为非阻塞型。与内网的连接套接字存入用于 sockets2 数组，循环查找 sockets2 数组中的闲置位置（allclosed [j] =true 连接全关闭），将其置入，修改对应的状态标志，设定 unused [index] = true; 转发对变为半连接待用状态。

3.6 外网客户端请求处理模块 accept_local ()

accept_local () 处理来自于外网主机上的客户端请求，代码如下：

```
void DataModule::accept_local()
{
    struct sockaddr_in client;
    int size=sizeof(struct sockaddr);
    SOCKET nfd = accept(lolisten, (struct sockaddr*)&client,
&size);
```

```
if (nfd == INVALID_SOCKET)
    return;
ioctlsocket(nfd, FIONBIO, &j); //设定为非阻塞套接字
int index = -1;
for (int  j = 0; j < SOCK_SIZE; j++) {
    if (unused[j]) {
        index = j;
        break;
    }
}
if (index == -1)    {
    closesocket(nfd);
    return;
}
sockets1[index] = nfd;
unused[index]  =  false;
s1closed[index]  = false;
  connectsum--;   //与内网待用连接数量递减
ctrlbuf[0]='+';
if( connectsum < 5 )  //当待用连接数小于 5 时,通知内网
//增加待用连接数量
        send( ctrlsocket,ctrlbuf,16,0);

}
```

当与本地的连接套接字 nfd 建立后，循环查找套接字状态数组 unused [index] =true 的位置，表明 index 位置上，连接内网套接字 sockets2 [index] 已建立，将 nfd 置入连接本地套接字数组 sockets1 [index] = nfd，至此，一对套接字转发对已经完全建立，可以在这两个套接字上转发数据了。

connectsum 表示与内网待用连接总数，每建立一个转发对，待用连接就要减少，当 connectsum 减少到 5 时，通过 ctrlsocket 通知内网转发器，再次建立一定数量的连接以备用。

3.7 数据转发过程

客户端请求数据以及内网应答数据的转发原理，概括的说，转发对中一个套接字接收的数据，由另一个套接字发送出去。具体过程为：与本地客户端连接的套接字 s1sockets [i] ,接收由客户端传来的请求数据，将数据保存在字节缓冲区 sendbuffer [i] ,同时记录请求数据长度于 s1recvbyte [i]，然后由连接内网转发器的套接字 s2sockets [i]，把存放于缓冲区 sendBuffer [i] 中的数据发送给内网转接器。与此过程同理，与内网转发器连接的套接字 s2sockets [i] ,接收由内网转发器传来的应答数据，保存在缓冲区 recvbuffer [i] ,同时记录应答数据长度于 s2recvbyte [i]，然后由套接字 s1sockets [i]，把存放于缓冲区 recvbuffer [i] 中的应答数据传给本地客户端。无论接收和发送，都要记录实际成功完成的字节数，并相应移动缓冲区指针，以防止未发送的数据被后到数据覆盖，或者重复发送数据。

函数 recv_from_socket1 完成接收本地客户端请求数据的功能，具体代码如下：

```
void DataModule:: recv_from_socket1(int i)
{
    int ret = recv(sockets1[i], sendbuffer[i] + s1recvbyte[i],
BUFSIZE - s1recvbyte[i], 0);
    if (ret == 0) {
        handle_close_from_socket1(i);
        return;
    }
    if (ret < 0) {
if (GetLastError() == WSAEWOULDBLOCK ||  GetLastError()
== WSAEINPROGRESS)
            return;
        handle_close_from_socket1(i);
        return;
    }
    s1recvbyte[i] += ret;
}
```

recv_from_socket1 (int i) 有一个参数，指明被调用的套接字数组索引。该函数首先调用 winsock 接收函数 recv，第一个参数为连接本地客户端套接字 sockets1 [i] ,第二个参数为接收缓冲区（字符类型的数组）首地址，因为上一次接收的请求数据可能还没有被与 sockets1 [i] 对应的 sockets2 [i] 发送完毕，所以新接收的数据要从上次接收数据的末尾位置开始存放，这样接收缓冲区首地址指针要偏移到 sendbuffer [i] + s1recvbyte [i] 的位置；第三个参数为接收缓冲区长度，也因为上一次接收数据可能尚未发送完毕，缓冲区长度为缓冲区总长减去已接收数据长度 BUFSIZE - s1recvbyte [i]。第四个参数指定调用方式，一般为 0。

对 recv 返回值 ret 的判断是本函数的关键点。当 ret 为 0 时，表明 sockets1 [i] 连接已断开，流程转入 handle_close_from_losockets1 (i) 处理断开连接，当 ret<0 时，说明有错误发生，先调用函数 GetLastError () 获取错误代码，若错误代码值为 WSAEWOULDBLOCK，表明套接字上无数据可读，但连接还是正常的，返回值为 WSAEINPROGRESS，表明套接字上数据处理仍在进行，尚未完成。这两种情况函数不做任何处理，直接返回，其他错误均按 sockets1 [i] 断开处理。如果 ret>0，表明已正常接收到数据，ret 值即为接收数据长度，累加已接收数据长度值：s1recvbyte [i] += ret。

函数 send_to_socket2 (int index)，把 recv_from_socket1 (int index) 函数接收的请求数据，发送给内网。代码如下：

```
void DataModule:: send_to_socket2 (int i)
{
  if (closing[i] && (s2sentbyte[i] == s1recvbyte[i])) {
        s2closed[i] = true;
        allclosed[i] = true;
      closesocket(sockets2[i]);
        return;
```

```
    }
    int ret = send(sockets2[i], sendbuffer[i] + s2sentbyte[i],
s1recvbyte[i] – s2sentbyte[i], 0);
    if (ret < 0) {
if (GetLastError() == WSAEWOULDBLOCK || GetLastError()
== WSAEINPROGRESS) {
            return;
          handle_close_from_socket2(i);
          return;
  }
  s2sentbyte[i] += ret;
  if (s2sentbyte[i] == s1recvbyte[i]) {
          s2sentbyte[i] = 0;
          s1recvbyte[i] = 0;
  }
}
```

函数在发送数据之前，先对 SOCEKT 的半关闭状态进行判断。closing [i] 为真，说明 sockets1 [i] 已经关闭，s2sentbyte [i] == s1recvbyte [i] 说明请求数据发送长度等于请求数据接收长度，请求数据已全部发送完毕，sockets2 [i] 也可以关闭了。整个转发对进入全关闭状态。

调用 winsock 函数 send 发送数据，send 函数将请求数据发送给 sockets2 [i] 的对端，第二个参数为接收数据缓冲区首地址 sendbuffer [i] + 已发送数据长度 s2sentbyte [i]，含义是未发送数据的首地址，第三个参数指明发送缓冲区长度，等于已接受的字节数减去已发送字节数，即为未发送数据的长度。第四个参数一般置 0。判断 send 返回值 ret，若 ret<0，处理过程同于recv_from_socket1 ()，若返回值>=0 则发送成功，将发送字节数累加 s2sentbyte [i] += ret 。

最后判断，若 s2sentbyte [i] == s1recvbyte [i]，表明接收的请求数据已全部发送完毕，两个值重新归零，缓冲区 sendbuffer [i] 重新从首位置开始接收数据。整个过程没有数据的转移、拷贝，效率较高。

recv_from_socket2 和 send_to_socket1 原理与上述函数类似，不再赘述。

recv_from_socket2 从内网接收应答数据，代码如下：

```
void DataModule:: recv_from_socket2(int i)
{
  int ret = recv(sockets2[i], recvbuffer[i] + s2recvbyte[i],
BUFSIZE – s2recvbyte[i], 0);
    if (ret == 0) {
        handle_close_from_socket2(i);
        return;
    }
    if (ret < 0) {
            if (GetLastError() == WSAEWOULDBLOCK ||
GetLastError() == WSAEINPROGRESS)
            return;
        handle_close_from_socket2(i);
        return;
    }
    s2recvbyte[i] += ret;
}
```

函数 send_to_socket1 把从内网接收的应答数据发送给本地客户端：

```
void DataModule::send_to_socket1(int i)
{
  if (closing[i] && (s1sentbyte[i] == s2recvbyte[i])) {
        s2closed[i] = true;
        allclosed[i] = true;
      closesocket(sockets1[i]);
        return;
    }
    int ret = send(sockets1[i], recvbuffer[i] + s1sentbyte[i],
        s2recvbyte[i] – s1sentbyte[i], 0);
    if (ret < 0) {
            if (GetLastError() == WSAEWOULDBLOCK ||
GetLastError() == WSAEINPROGRESS)
            return;
        handle_close_from_socket2(i);
        return;
    }
    s1sentbyte[i] += ret;
  if (s1sentbyte[i] == s2recvbyte[i]) {
    s1sentbyte[i] = 0;
    s2recvbyte[i] = 0;
    }
}
```

3.8 转发对套接字关闭处理

转发对中套接字的关闭由函数 handle_close_from_socket 处理，代码如下：

```
void DataModule::handle_close_from_socket1(int i)
{
  closing[i] = true;
  closesocket(sockets1[i]); //关闭套接字 sockets1[i]
    s1closed[i] = 1;
    if (! s2closed[i]) {
    shutdown( sockets2[i],SD_RECEIVE); //不允许 sockets2[i]
//再接收数据
    }
}
```

转发对有一个套接字关闭，则整个转发对就失效了，与 sockets1 [i] 配对的 sockets2 [i] 已无需再接收数据，但此时由 sockets1 [i] 已接收的数据可能尚未发送完毕，那么 sockets2 [i] 还需要继续发送数据，对 sockets2 [i] 采取半关闭策略，调用 winsock 函数：

```
shutdown( sockets2[i],SD_RECEIVE);
```

该函数第 2 个参数有 3 选项，SD_RECEIVE，表示不允许再对此套接字调用接收函数。对于 TCP 套接字来说，无论数据是在等候接收还是即将抵达，都要重置连接。SD_SEND，不允许再调用发送函数。对于 TCP 套接字，会在所有数据发送出并得到接受端确认后产生一个 FIN 包。SD_BOTH，则包含前两个选项的功能。第二个参数取 SD_RECEIVE，只关闭接收，sockets2 [i] 仍可继续发送剩余数据。

3.9 心跳包机制维持连接的有效性

局域网内部向外网的连接置于网关 NAT 的管理下，如果持续一段时间该连接上无数据传输，NAT 会将其关闭而使连接失效。因此需要定时发送一个数据包，称之为心跳包，使连接保活。TCP 协议在底层实现了定时传送一段探测包，侦测对方是否关闭的功能，函数名为 WSAIoctl，可以对某个套接字单独设定，由 TCP 协议栈负责定时发送，不影响上层数据传输。探测包实质上起到心跳包的功能：

```
bool SetSocketKeepAlive( SOCKET m_sock, int second)
//设置 KeepAlive
{
    tcp_keepalive in = {0}; //输入参数
    tcp_keepalive out = {0}; //输出参数
    unsigned long ulBytesReturn = 0;
    in.onoff = 1;
   in.keepaliveinterval = 1000*3; //连接不畅通时的测试频率
    in.keepalivetime = 1000*second; //表示的是 TCP 连接处
//于畅通时候的探测频率
int nRet=WSAIoctl(m_sock, SIO_KEEPALIVE_VALS, (LPVOID)
&inKeepAlive,sizeof(in),(LPVOID)&out, sizeof(out), &ulBytes
Return, NULL, NULL);
}
```

keepalivetime 表示的是 TCP 连接处于畅通时候的探测频率，一旦探测包没有返回，就以 keepaliveinterval 的频率发送，经过若干次 (2000 XP 2003 默认 5 次，Vista 默认 10 次) 的重试，如果探测包都没有返回，说明 TCP 连接已经断开。心跳包发送由外网转发器负责，可以减轻内网转发服务器的负担。

4 结语

系统的优势在于无需在网关上作任何设置，安装任何辅助软件，就能实现公网上连接内网主机服务的功能，转发系统有用户认证，确保足够安全的条件下访问内网。系统已成功运用于远程办公、远程管理内网等场合。

参考文献

[1] [美] Gray R Wright，Richard Stevens. TCP/IP 详解 [M] . 陆雪萤，等，译. 北京，机械工业出版社，2000.

[2] Hosner C. Open VPN and the SSL VPN Revolution [J] .Sans Institute InfoSec Reading Room,2004.

[3] 罗红，幕德俊，等. Research on Communication Techniques for Port Recall Trojan Horse [J] . 微电子学与计算机，2006.

（收稿日期：2013-02-25）

王文京：小微企业云服务落地

随着以移动互联网、云计算、大数据为代表的新技术的迅猛发展，企业管理和经营模式互联网化，企业生产和经营活动对移动互联网和云计算的依赖愈发加强。用友集团董事长王文京指出，我们将继续加强平台化发展、产业链共赢，顺应全球 IT 产业的革新浪潮。在小微企业层面，我们将结合新技术加快小微企业信息化软件与服务发展，打造新一代小微企业云计算服务平台，创新小微企业信息化新模式，引领小微企业进入新型信息化时代，让小微企业享受到真正落地的云服务。

小微企业信息化需求转变

云计算以及互联网的快速发展，引发小微企业生意模式、管理模式和工作模式变革，以流程、规范、控制为核心的管理软件，已经不能满足企业内部高效协同、产业链协同、采用各种新兴营销手段拓展经营的需要。单纯企业管理将向企业内部管理与经营管理、外部市场管理、社会资源管理发展，管理信息化向经营信息化转变。

小微企业信息化新特点

小微企业新型信息化以管理与经营并重，高度依赖云平台和云服务支撑，让企业可以更便捷的获得经营发展所需要的各种社会资源，快速决策，加快发展。王文京表示，软件企业在技术上必须利用云计算和互联网向用户提供更为全面和灵活的信息化服务，满足企业日常经营需要。

2011 年 3 月，畅捷通发布了以“畅捷通 T+”为核心的 S+S（云+端）新业务发展策略，通过整合云服务帮助小微企业在使用软件进行信息化管理的同时，利用云服务快速获取企业经营信息，加速信息传递和决策制定，增强应对市场变化的能力，实现企业低成本、高效率、快发展。

2013 年 4 月，畅捷通将正式发布专为小微企业量身打造的云时代的企业运营平台“畅捷通 T+”，“畅捷通 T+”综合云时代移动互联、社交网络、云计算三大技术特性，满足企业内部管理、协同工作、业务拓展的整体运营需要，是企业全新的信息化服务平台，将让企业更加高效的利用云计算为经营信息化服务，让云服务在小微企业落地。

使用“维棠”轻松完成 FLV 视频下载与转换

马凤娟 吴鹏飞

摘 要：“维棠”是一款专门针对 FLV 格式的视频下载而开发的软件，用户使用“维棠”可以将优酷网、土豆网、酷 6 网、56 网等网站上的 FLV 视频节目轻松下载到自己的电脑上保存下来，并可以根据需要转换成不同的格式。

关键词：维棠；FLV 文件；下载；格式转换

1 FLV 视频简介

FLV 视频即 Flash Video，是 Macromedia 公司推出的一种流媒体视频格式，与其他视频格式相比，FLV 视频文件较小，加载速度非常快，便于在互联网上传输。并且，FLV 视频采用嵌入在浏览器中的 Flash Player 来播放，用户不需要再安装另外的视频播放器就可以直接观看，非常方便。因此，FLV 格式推出不久，就受到各大网站的青睐，例如优酷网、土豆网、酷 6 网、56 网等都采用 FLV 作为自己的网络视频格式，内容上涵盖娱乐、教育、新闻等方面，一时间，互联网上出现了大量的 FLV 文件。

但是大家在使用的过程中或许已经发现，这种 FLV 的视频格式，只能在线观看不能下载，即便是下载也要借助于该网站提供的下载软件，例如 56 网提供的 56ican、土豆网提供的 iTudou 等，播放这些文件也需要专门的播放器，这显然是不方便的，而且安装这么多的插件也会加重电脑的负担。只安装一个“维棠”，就可以轻松地解决各大网站的 FLV 格式视频的下载、播放及格式转换问题，接下来就一起来看怎么做吧。

2 安装“维棠”软件

2.1 下载

首先从“维棠”官网（网址：http://www.vidown.cn/supportlist.html）下载该软件，目前最新版本是ViDown_1.1.2.2。

2.2 安装

（1）双击 ViDown_1.1.2.2_setup.exe 进行安装，将会弹出一个选择安装语言的对话框，选择“简体中文”，单击“OK”按钮。

（2）从接下来弹出的对话框中单击“下一步”按钮。

（3）然后从弹出的对话框中单击“我接受”按钮，接受授权协议中的条款。

（4）然后会弹出选择安装目录的对话框，单击“浏览”按钮可以选择自己要安装的目录。按默认设置，单击“下一步”按钮继续安装。

（5）接下来的对话框主要是一些附加任务的选择，例如是否创建桌面图标、快速启动图标等，可以根据自己的需要进行选择某些任务或取消某些任务。然后单击“安装”按钮，开始安装。

（6）稍微等待几秒钟即可，从弹出的对话框中单击“完成”按钮，即可完成安装。

（7）安装完成后将自动运行该软件，或者从“开始”菜单中选择“程序”→“维棠 FLV 视频下载软件”，或者单击桌面快捷图标，都可以启动该软件。启动后界面如图 1 所示。

图 1 程序运行界面

该软件的界面比较简单，上方是该软件的菜单栏和常用的工具栏，主要的操作都是通过菜单栏和工具栏里的命令来完成的。左侧是一个列表，包括正在下载的视频、已经完成下载的视频和已经删除的视频列表，需要查看哪一部分，直接单击选择即可。

3 使用“维棠”下载 FLV 视频

在优酷网、土豆网等网站浏览视频时，如果想下载到自己

（下转第 93 页）

TROUBLESHOOTING OF PROGRAM

Q VBA 中常见自定义控件应用有何技巧

A 不少朋友喜欢对 Office 进行二次开发，用 VBA 来提高自己的办公效率，恰当、合理地使用 VBA 提供的自定义控件不仅可以让应用程序窗体变得更加美观，同时也能使设计的应用更加的个性化、更加方便用户的使用。

在默认的情况下，窗体控件工具箱中仅给出了常见的几个控件，事实上，VBA 的控件数量远不止如此。用户可以根据需要在控件工具箱上添加一些标准控件或自定义控件，也可以随时删除它们。接下来，为大家介绍几个常见的自定义控件的使用方法，希望大家能充分地运用这些控件，这样会使应用程序增色不少。

(1) 日历控件 (Calendar)

在 Excel VBA 窗体的工具箱中没有提供日历控件对象，事实上很多应用程序在设计的时候需要用到此控件，利用它可以方便地插入用户输入的日期。首次使用该控件时需要将其添加到窗体工具箱中。添加的方法是：

打开 VBE 编辑器，首先在当前工作薄中插入一个用户窗体。接下来单击“工具”菜单的“附加控件”命令，打开“可用控件”对话框，在“可用控件”列表框中选择“日历控件 11.0”（Office 版本为 2003），单击“确定”按钮。此时在窗体工具箱中多出了一个“Calendar”（日历）控件。

与其他可编程控件一样，日历也有很多的属性，如：BackColor 属性可用来设置日历控件的背景色；Year、Month 和 Day 属性可用来提取所选日历的年份、月份和日期；Value 属性用于显示用户在日历控件中选定的日期值等。

接下来，看一个关于日历控件应用的例子：

当在工作表中输入出生日期时，系统自动弹出插入日期的窗体，用户选择日期并确认后，日期将被插入到指定的单元格中。

打开 VBE 编辑器，在当前工作簿中插入一个用户窗体，利用日历控件和按钮控件设计如图 1 所示的用户界面。双击窗体中的命令按钮，在其 Click 事件中编写下面的代码：

```
Private Sub CommandButton1_Click()
ActiveCell.Value = Calendar1.Value
UserForm1.Hide
End Sub
```

双击当前的工作表（如 sheet1），在其 SelectionChange 事件中编写下面的代码：

```
Private Sub Worksheet_SelectionChange (ByVal Target As Range)
'判断用户选中的是否为第 3 列
If Target.Column = 3 Then
UserForm1.Show
End If
End Sub
```

输入数据时，当用户将光标移到到第 3 列即出生日期所在列时自动弹出用户窗体，用户选择完日期并单击“插入日期”按钮后，所选日期自动插入到当前的单元格中，同时用户窗体自动隐藏。这样，用户就可以方便地录入日期数据了。

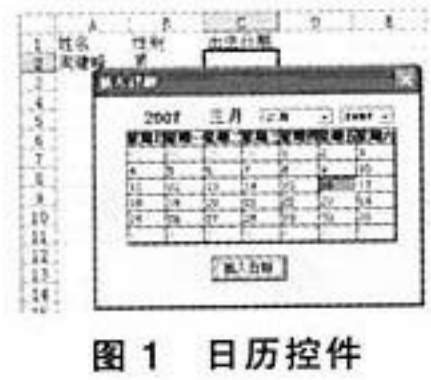


图 1 日历控件

(2) 日期控件 (DTPicker)

与日历控件一样，日期控件也是一个用来获取日期的控件，功能大致相同，但操作界面与日历控件略有区别，该控件在应用程序的开发中比日历控件应用更为常见。

首次使用日期控件时也需要将其添加到窗体工具箱中去，添加的方法与日历控件类似，日期控件的名字是“Microsoft Date and Time Picker Control 6.0 (SP4)”。其主要属性有：Format 属性用于指定日期的显示格式；Value 属性值与用户选定的日期值有关；Year、Month 和 Day 属性可用来提取所选日期值的年份、月份和日期等。

也来看一个关于日期控件使用的例子：

使用窗体查询或采集数据时经常会遇到日期输入的问题。此时既可以手工输入日期数据，也可以单击列表框右侧的下拉按钮选择日期数据。图 2 为日期控件在应用程序设计中的典型应用，系统可将用户选择的日期值回显到指定的文本框或其他对象中，从而方便了日期数据的查询或采集。

图 2 日期控件

(3) Flash 控件

在 Office 组件中利用控件工具箱提供的“其他控件”中的“Shockwave Flash Object”对象可以在文件中插入 Flash 文件，其实在 VBA 的应用程序窗体中也可以插入 Flash 对象。要使用 Flash 控件，首先需要从“附加控件”对话框中选中它并将其添加到窗体工具箱上，该控件的名字是“Shockwave Flash Object”。

在窗体中添加 Flash 对象的方法与在文档中添加 Flash 的方法相同，这里就不再赘述了。利用 Flash 控件对象可以制作出精美的欢迎界面、功能强大的 CAI（计算机辅助教学）软件以及一些个性化的应用程序菜单等，结合 Flash 本身强大的 Action Script 命令，几乎无所不能。

(4) 进度条控件 (ProgressBar)

进度条控件是用来显示进度的控件，该控件在应用程序的加载、下载等待、安装等应用中非常的广泛。要使用进度条控件，首先需要从“附加控件”对话框中选中它并将其添加到窗体工具箱上，进度条控件的名字是“Microsoft ProgressBar Control 6.0 (SP4)”。

该控件的主要属性有：Max 属性用于指定进度条控件的最大值；Min 属性用于指定进度条控件的最小值；Value 属性用于显示进度条的进度值。

下面例子可以实现模拟的应用程序进度加载效果。利用进度条控件和标签控件，可以设计一个如图 3 所示的窗体界面，最后在窗体 Activate 事件中编写如下代码就可以模拟其效果了。

```
Private Sub UserForm_Activate()
ProgressBar1.Min = 1
ProgressBar1.Max = 20000
Label2.Caption = "0%"
For i = 1 To 10000 Step 0.1
ProgressBar1.Value = Str(i)
Label2.Caption = Round(i / 10000, 2) * 100 & "%"
DoEvents
Next i
End Sub
```

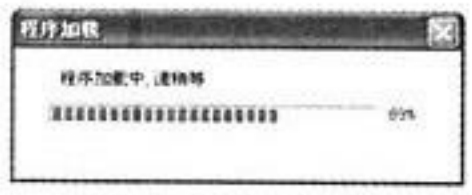


图 3 进度条控件

朋友们可以根据演示的速度来调整代码中的相关数值。需要说明的是：代码中使用了 DoEvents 函数，其作用是转让控制权，以便让操作系统处理其它的事件。当需要执行一段循环代码时，特别是循环的次数较大时，为了防止出现死机或死循环，在循环代码的开始加上 DoEvents 函数以便在程序运行过程中可以中止程序的运行。本例稍有遗憾的是无法根据系统的运行情况实现进度条加载速度快慢的控制，感兴趣的朋友可进一步做些尝试。

其实 VBA 中的自定义控件也不止今天介绍的这几个，在很多时候，其实用户的很多需求利用这些自定义控件就可以很方便地实现了，而不是需要辛辛苦苦地去重复编写这些代码，当然这需要大家平时就要留意收集和整理这方面的技巧与心得。

（作者：仲勇）

Q 质数判别有哪些问题有待探讨

A 质数在研究整数的过程中占有一个很重的地位，它被称为自然数的“建筑的基石”。虽然有很多数学家和学者致力于对它的研究，但成果并不显著，仍有许多问题有待解决。例如，哥德巴赫猜想困扰了人们几百年，有很多数学家对它进行了多年的研究但并没有得到解决。随着现代科学技术的迅速发展，运用计算机能够较快地计算某自然数是否是质数，知道在某范围内质数的分布情况。另外。质数在信息学奥林匹克竞赛中也屡屡出现，技巧性非常强，可以锻炼和提高学生的思维，所以有必要对质数在计算机上如何进行判别的相关问题进行一些探讨。

(1) 质数的定义法

1) 如何判别一个数为质数

什么是质数？质数是一个大于 1 的整数，如果它的正因数只有 1 及它本身，这个数就叫质数（或素数）。通俗的讲，质数是只有 1 和它本身两个约数的自然数，不能被任何其他的自然数整除。

如何判别一个数是质数呢？按照上述对质数的定义，就要用 2 到这个数减 1 的数分别去除这个数，如果都不能把它整除了，那么这个数就被认为是质数。用程序表示如下：

```
var
  n, i: integer ;
  f:boolean;
begin
  read(n);
  f:=true;
  for i:=2 to n-1 do
    if n mod i=0 then  f:=false;
 if  f  then writeln('YES')
     else writeln("NO');
end.
```

这个程序按照质数的定义来判断一个整数是否是质数是没什么问题的，但它还是做了许多无用功。比如，来判断 15 是否为质数。如果用上面的程序，就得从 2 到 14 分别去除 15，其实是没有必要的。如果 15 能被 3 整除了，就说明它是合数了，还有必要再用 4~14 去除吗？这时就应中断循环。另外，在循环的次数上还可以优化，假如判定 36 是否为质数，只要从 2 到 6 分别去除 36 就可以了，为什么？因为：2×18=36；3×12=36；4×9=36；6×6=36；如果 2、3、4、6 能把 36 整除了，那 18、12、9 肯定也能把 36 整除。所以可把上面的程序优化为：

```
var
  n, i:integer ;
  f:boolean;
begin
  read(n);
  f:=true;
  for i:=2 to round(sqrt(n)) do
    if n mod i=0 then  begin
                  f:=false;
                  break;
                  end;
 if  f  then writeln('YES')
```

```
        else writeln('NO');
    end.
```

经过优化后的程序，大大缩短了质数的判别时间。

2）如何找出任意两个正整数之间的质数

如果再寻找某一区间内的所有质数就好办了，只要在上面的程序加上一个大循环就行了。如，找出 a~b (a≤b,a,b 是正整数) 之间的质数，实现的程序就是：

```
var
  a, b, n, i:integer ;
  f:boolean;
begin
  read(a,b);
   for n:=a to b do
    begin
      f:=true;
         for i:=2 to round(sqrt(n)) do
            if n mod i=0 then  begin
                        f:=false;
                        break;
                        end;
        if  f  then write (n,'  ')
 end;
end.
```

上述程序对整型（integer）内的数进行判定一般不成什么问题，一旦数大了，会出现超时。尤其是奥林匹克竞赛上的试题，测试数据有的是很大的，如果超时就难获得满分，就得采用另一种方法——筛选法。

(2) 合数过滤筛选法

质数是不能被 2~（n-1）间的任何数整除；反过来看，只要能被 2~（n-1）间的任何数整除的 n，都不是质数。我们可以采用一个简单的排除法：就是对 n 以内的所有数，只要逐个去掉值为 2~（n-1）的倍数的数，剩下的数就是质数。

现在以一个奥赛试题进行分析：选数（2002 年试题）：

问题描述：已知 n 个整数，x1，x2，x3，……xn，以及一个整数 k (k<n)。从 n 整数中任选 k 个整数组合相加，可分别得到一系列的和。现在要求你计算出和为素数（质数）的组合数有多少种。（1≤n≤20，k<n，1≤xi≤5000000 ）。在此只对试题质数的判定来分析：由于整数 xi 的上限为 5000000，k 的上限为 19，这就使得判别 k 个整数的和是否为质数的问题似乎有点困难。为了保证在该范围内能正确出解，且不超时，先进行筛选，将 1~10000 间的质数筛选出来存入质数表 P 中。由于 P 表中的最大质数接近 10000，其平方大于或等于 xi 和的上限 19×5000000 (k 最大值是 19)，因此是一个比较可行的方法：

```
var
  f:array[1..10000] of boolean; {标记是否为质数}
  p:array[1..5000] of longint;  {存放质数}
  k:integer ;
```

编写一个过程来筛选质数：

```
Procedure  prime;
 Var
   i, j : integer;
 begin
    k:=0;
    fillchar(f, sizeof(f),true);
    f[1]:=false;
    for  i:=2  to  10000  do
    if  f[i] then begin
              inc(k); {计质数的个数}
              p[k]:=i; {存放质数}
              j:=i+i;{能用加法就不要用乘法}
              while j<=10000 do  {将 i 的倍数筛掉}
                 begin
                   f[j]:=false;
                   j:=j+i;
                 end;
            end;
 end;
```

上面的程序运行后，在质数表 P 中就有 k 个质数。在质数表 P 的基础上再去判别和数 sum 是否为质数。大家知道，任何一个大于 1 的整数都可以分解成质因子的乘积形式，而 P 表中的质数是按递增方向存放的，如果用 P 表中的每个质数去除 sum，若某个质数能把 sum 整除，则说明 sum 为合数；若 P [i] *P [i] >sum，则说明 sum 不可能分解出比 P [i] 大的质数了，sum 本身就是质数了。判别函数程序如下：

```
function  text(sum:longint):boolean;
 Var  i:integer;
 begin
    text:=true;
    for i:=1 to k do
    begin
      if  p[i]*p[i]>sum then break {若超出搜索范围的上限,则说明 sum 是质数,返回 true}
      else  if sum mod p[i]=0 then begin
                              text:=false;
                              break;
      end; {说明 sum 不是质数,返回 false}
    end;
 end;
```

这样在程序中只要调用一次过程，就可将质数筛选出来，再将每次组合的和数用函数去判别，大大提高了程序的运行时间。

再看一个实例，大家都听说过哥德巴赫猜想的问题，迄今为止，这仍然是一个著名的世界难题，被誉为数学王冠上的明珠。

问题描述：任何一个大于 2 且不超过 n 的偶数都能写成两个质数之和。

问题分析：由题意可知，程序是要对组成上述范围内的任一个偶数的两个整数进行质数判别。如果 n 的数不是很大，用

2 到 round (sqrt (n)) 之间数去除，进行两次判别，机器还能承受；如果 n 的值很大，肯定是要超时的。为了保证 n 是大数时能出正确解，是用筛选法来编写程序，验证哥德巴赫猜想问题。（在这里 n 是大于 2 且小于等于 10000 之间的偶数）：

```
var
  n,i:longint;
  m,j,k:integer;
  p:array[1..10000] of longint;
  f:boolean;
begin
  readln(n);
  m:=1;
  p[1]:=2; {将质数 2 放入 P 表中}
  for i:=3 to n do {筛选 n 以内的所有质数}
   begin
    f:=true;
    for j:=1 to m do {用质数表中的质数去除 i,如果整除 i 就是合数}
     if i mod p[j]=0 then  begin
                            f:=false;
                            break;
                           end;
    if f then {如果不能被整除,i 就是质数并放入 P 表}
     begin
      inc(m);
      p[m]:=i;
     end;
   end;
  for i:=2 to (n div 2) do {验证任一个大于 2 的偶数都可以用两个质数和表示}
    for j:=1 to m do
     begin
      f:=false;
      for k:=j to m do
       if i+i=p[j]+p[k] then
        begin
         f:=true;
         writeln(i+i,'=',p[j],'+',p[k]);
         break;
        end;
       if f then break;
     end;
end.
```

这个问题还是很有趣的，同学们不妨运手试一试。

(3) 其他方法与技巧

有些问题中，并不是涉及到质数都得要去判别是否为质数，只要算法合适，问题完全可以简化。比如：求 2~100 中，每个数的质因子。输出形式如下：

```
2=2
3=3
4=2*2
5=5
6=2*3
.....
100=2*2*5*5
```

认真观察，仔细分析上面的输出形式：发现它是先用最小的质数 2 连续去整除，直到不能被整除了，如果这时商不为 1，将除数加 1，继续去除，直至商等于 1 时，说明该数分解完成。由于在每次开始时都是用质数连续去除，直到不能整除为止，所以出现的因子都会是质因子。程序如下：

```
Program  zhiyinzi;
  var  a, b, n: integer;
 begin
  for n:=2 to 100 do
   begin
    a:=2;
    b:=n;
    write(b,'=');
    repeat  {b 是变化的被除数,a 是变化的除数}
  while b mod a<>0 do   a:=a+1;  {b 不能被整除,除数加 1}
     b:=b div a; {b 被 a 整除了,产生新的被除数 b}
     if b=1 then write(a) {b=1 时分解完毕}
          else  write(a,'*'); {b<>1 时,输出当前质因子 a,等待下一个质因子的输出}
    until b=1; {本次分解质因子结束}
    writeln;
   end;
end.
```

从这个程序中可以得知：任何一个大于 1 的正整数都可以用质因子的乘积表示。

看一下 2012 年普及组复赛的第一题“质因数分解”。

问题描述：已知正整数 n 是两个不同的质数的乘积，试求出较大的那个质数。

输入文件：只有一个正整数 n；

输出文件：较大的那个质数 p；

数据范围：6≤n≤2×109。

问题分析：根据给出的数据范围，n 的上限达到 2×109，如果它的两个因子都要判别是否为质数的话，那一定会超时的。由前面的例题已经知道：任何一个正整数都能表示成质因子的乘积。而此题正是 n 是两个不同的质数的乘积，那给出的 n 就不会出现像 7、28 这样的数（7=7，28=2×14，28=4×7）。那只要在 2 到 trunc (sqrt (n)) 范围内找出能把 n 整除的数，这个数就是那个较小的质因子，较大的那个质因子也就迎刃而解了。程序如下：

```
program prime;
 var  n,i:longint;
 begin
  readln(n);
```

TROUBLESHOOTING OF PROGRAM

```
  for i:=2 to trunc(sqrt(n)) do
    if n mod i=0 then break;
  writeln(n div i);
end.
```

此程序全部通过给出的考试测试数据。

通过以上分析，大家对质数判别问题可能有了初步的认识。若能够掌握其算法思想对解决实际问题还是很有必要的。在平时的训练和学习中，注意锻炼思维能力，勤于思考，发现规律，在面临一个个题型时能够快捷准确地建立模型，确定算法，解决问题。以上仅是个人的理解，希望和大家共勉。

（作者：檀素芳）

（上接第 88 页）

电脑上保存下来，只需要启动“维棠”，单击工具栏中的新建按钮（如图 2 所示）。

图 2　新建下载

单击新建按钮后，打开如图 3 所示的页面，将想要下载的视频所在页面的网址拷贝到“视频网址”后面的对话框中，单击“另存到”后面的节略号，指定视频要存储的位置。然后设置一些参数或按照默认设置，单击确定，即可开始下载视频，如图 4 所示。

图 3　下载设置

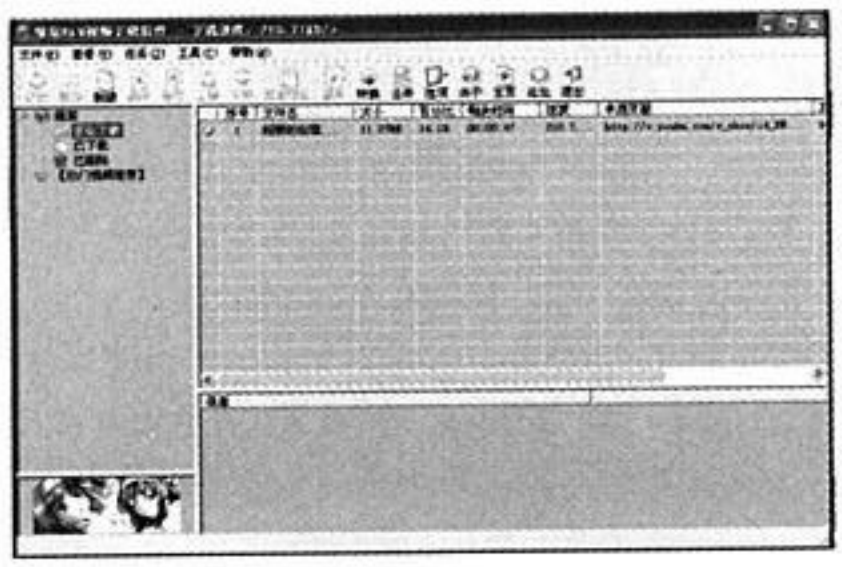


图 4　下载列表

4　FLV 视频格式转换

目前很多视频播放软件和课件制作软件还不支持 FLV 格式的视频，所以需要把下载的 FLV 文件转换成其他通用的视频格式，如 Avi、Wmv、Mp3、Mp4 等。做法如下：

（1）单击“工具”菜单栏，选择“转换管理器”，或者按 Alt+C 快捷键，打开如图 5 所示的对话框。

图 5　视频转换对话框

（2）单击图 5 中的“新建任务”按钮，打开图 6 所示的页面。从中单击“源文件”后面的“浏览”按钮，选择要转换的源文件，单击“输出文件”后面的“浏览”按钮，指定要输出文件的保存路径和要输出的格式，单击“确定”按钮，即可完成格式转换。操作非常方便，也可以根据需要截取视频中的某个片段进行转换。

图 6　转换设置

这样只需要一个软件“维棠”，就可以完成下载、格式转换的功能，完成后可以得到 Avi、Wmv、Mp3 等格式的视频，可以直接用于 MP3、手机等其他的播放设备，也可以插入到 Powerpoint 等软件中作为演示材料，这一点对于教师朋友们有很大的帮助。

（收稿日期：2012-12-17）

电脑系统维护经验与技巧

如何用 Windows 7 自带的功能分区

Windows 7 相对于旧版的 Windows，它的功能有所增强，它依靠自身而不依赖第三方就能无损分区。点击“开始”，用鼠标右键点击计算机，在弹出的菜单中选择“管理”，然后依次展开“计算机管理（本地）→存储→磁盘管理”，右键点击“C 盘”，在弹出的菜单中选择“压缩卷”，分出要划分出来的分区，然后再在这个分区右键选择扩展卷划分几个分区即可。

如何正确使用西数绿盘

西数绿盘具备 Advanced Format 技术。所谓 Adanced Format 技术，其实是一种将硬盘扇区容量提高为 4KB 的技术。在传统的扇区分割机制中，每 512Byte 的数据之间，就需要间隔一个同步/分隔区域和一个 ECC 错误校验区域。而在 Advanced Format 模式下，每 4KB 才需要一个主样的扇区，相当于把以前的 8 个扇区合而为一，只需要一个同步/分隔区域和一个容量稍大的 ECC 校验区即可。目前来看，包括西部数据的 1TB、1.5 TB、2TB 这 3 大容量硬盘产品，均开始支持高级格式化技术。不过值得注意的是，该技术需要 Windows 7 等最新操作系统才能够完美地支持。如果仍使用 Windows XP 系统，则需要下载安装 WD Align 软件，才可以正常使用。

怎样针对指定文件夹碎片进行整理

对本本硬盘进行碎片整理，可以有效提高本本运行速度，但如今本本的硬盘容量越做越大，如果对对整个磁盘进行整理，速度会很慢。事实上，一般专门用于存放下载资料的文件夹碎碎片较多（特别是下载体积较大的视频文件），用户可以只对容易滋生碎片的文件夹进行整理，从而大大提高整理速度。

依次单击“文件→添加对象”，加入需要整理的文件夹或文件，可以加入多个，然后选中它们，单击“分析”，完成拍可看到碎片情况统计。如果觉得碎片较多的话，单击“整理→不检查磁盘”，结束后可以从“整理报告”中看到最终整理结果。

USB 接口可否转接为 IEEE 1394 接口

如果电脑主板上没有 IEEE 1394 接口，能不能买一根一端是 USB 接口一端是 IEEE 1394 接口的数据线连起来使用呢？答案是肯定的。因为 USB 接口与 IEEE 1394 接口是完全不同的两种接口，传输协议以及电压都不相同。现在市面上那种一端是 USB 接口，另一端是 IEEE 1394 接口的连接线，只是将里面的 4 个接点用线连起来而已，没有任何协议转换。使用这样的连接线时，电脑只会提示发现新硬件，但设备根本无法工作。并且由于 USB 接口里面的 4 根线中有一组是电源，而 DV 插头的 4 根线全是数据线，这样连接时极易将 DV 的 IEEE 1394 驱动电源烧毁，切记不要这样使用。最好还是购买一块 IEEE 1394 卡。

SATA 硬盘是否需要安装驱动

是否需要安装驱动主要看南桥芯片。如果主板上的南桥芯片型号为 VT8237，是必须安装驱动的。在 SATA 硬盘上安装系统时，需要按 F6 键，再加载 SATA 驱动。如果南桥芯片的型号为 VT8237R 或 VT8237R Plus，就不需要安装驱动了。为它们支持把 SATA 设备映射到 IDE 接口上，使用起来与 IDE 设备没有任何区别。另外，老主板在安装 SATA 硬盘时，最好升级为新版 BIOS，并下载官网上的 PDF 说明书进行设置操作，应该就没有多大问题了。

有哪几种方法不重装系统就可打开硬盘的 AHCI 模式

随着现在 CPU、显卡的性能越来越高，，硬盘性能已经成为影响电脑运行速度的瓶颈。在 Windows 7 系统中，通过开启硬盘的 AHCI 模式，可以在一定程度上提升硬盘的性能。但是很多用户的硬盘是在 BIOS 默认的 IDE 模式下安装了 Windows 7，后面更改硬盘为 AHCI 模式，就无法进行系统，必须重新安装系统才行。有两种办法不用重装系统，就能打开硬盘的 AHCI 模式。

（1） 方法一

先下载 MicrosoftFixit50470.msi 软件，运行后按照提示重新启动。按下 Del 键，进入 BIOS，在“Integrated Peripherals→RAID Config”界面中，将“SATA Operation Mode”选项更改为“AHCI”，不同主板 BIOS 的该选项名称有所不同。保存后重启电脑，这时再进入系统就不会自动重启了。

（2） 方法二（适用于 Intel 主板）

首先下载 Intel Matrix Storage Manager 和南桥芯片组 Matrix Storage Manager 驱动程序，将其解压缩后复制到 C:\Windows\System32\Drivers 文件内。在系统中按下 Win+R 键，调出“运行”对话框，键入“regedit”点击“确定”，打开注册表。进入“HKEY_LOCAL_MACHINE→System→CurrentControlSet→services→Msahci”选项，在右窗口中双击“Start”项目，会弹出一个对话框，将默认的参数由 3 更改为0。点击“确定”，并且保存，重新启动，进入 BIOS 设置界面，修改为 AHCI 模式即可。

在改好 BIOS 进入系统后，系统会自动安装 AHCI 的驱动

程序，完成后系统会要求再次启动，然后硬盘的 AHCI 模式就打开了。经过测试，硬盘在更改为 AHCI 模式后，在拷贝大文件时，速度有一定幅度的提升，而且硬盘在 Windows 7 的系统评分中，也提升了约 0.5 分，由此可见硬盘开启 AHCI 之后，效果还是不错的。

？ 怎样使用 DiskGenlus 工具软件直接进行各种硬盘格式转换

！ 在 Windows 7 操作系统中，只要某一硬盘分区大于是 32GB，那么在格式化磁盘时就只能将其格式化为 NTFS 文件系统，而移动硬盘或是 U 盘更是无法直接格式化成 FAT32 或是其他格式。尽管 NTFS 文件系统安全性较高，但还是存在各种对 FAT32 格式的需求（正如前面说的 PS3 和 DOS 的操作或是软件的研究和操作），DiskGenlus 的第一个强大的功能就是能直接进行各种硬盘格式的转换。

运行 DiskGenlus，在出现的主界面中，选择左侧驱动器列表的分区名称，并在其上单击鼠标右键，选择快捷菜单上的“格式化当前分区”选项。

在随后弹出的对话框内，展开“文件系统”列表，发现“FAT32”文件格式已经悄然出现。选择“FAT32”选项，按下“格式化”按钮，就能对所选分区进行格式化操作了。

此时返回 Windows 7 操作系统，双击桌面上的“计算机”图标，在刚才格式化的磁盘分区上单击鼠标右键，选择右键菜单中的“属性”选项，即可发现当前的磁盘分区格式已经成为 FAT32 格式了。此时就算用户重启系统，插拔硬盘或是 U 盘，再次连接它们时，依旧是 FAT32 系统，不会像 PQ10.0 那样出现反复显示 NTFS 系统的情况了。

？ 怎样使用 DiskGenlus 恢复文件

！ 例如恢复 U 盘中一个被删除的图片，进入 DiskGenlus 主界面，在左侧列表 U 盘盘符名称上单击鼠标右键，选择弹出菜单中的“已删除或格式化后的文件恢复”选项，在打开的对话框中点选“恢复误删除的文件”选项并同时勾选“恢复更早以前删除的文件”选项，按下“开始”按钮。待搜索文件的进度条结束后，即可得到被删除的文件列表。勾选希望恢复的文件，并在其上按下鼠标右键，从中选择相关的操作选项，便能够让误删除的文件“复活”了。

注意：当删除文件时（Shift+Delete 或清空回收站），系统只是在文件分配表做了小小的的修改——在该文件前面添加一个删除标志，其他文件可以随意占用它原有的空间。而进行数据恢复时，都是有限查询文件分配表，然后将删除标志去掉，让数据重新显示出来。如果数据保存区遭到破坏，那就很难修复了。

《VC 知识库视频大讲堂》成功上线

vckbase (VC 知识库) 网站成立于 1999 年，由温岭的王骏、广州的赵湘宁和安徽的赵涛凭借对 VC++的热爱创立起来的。王骏他们 3 个经常通宵达旦地工作，为站内添加了很多内容深刻的原创文章，广大网友为此被深深吸引。经过 2 年多的努力，到了 2013 年，vckbase 已经拥有 350 多万的注册用户。众多网友也纷纷来邮件投稿，有杨老师、徐景周、周星星等，他们为本站留下了极宝贵的财富。即便到了 10 多年后的今天，这些精彩的文章依然每天帮助着中国广大的 VC++程序员们。

很多软件企业的技术总监在感叹 C++程序员不好招,其实正是因为市场上缺少对 C++程序员进行系统培训的机构或组织。目前市场上的培训机构大多以 Java、.net、网站技术、手机开发等为主，而很少以 C++为主。原因是培训 Java 等非 C++方面的效益肯定要比培训 C++方面的要高出很多倍，培训机构是一个以盈利为目标的，所以寄希望于市场上培训机构来提升国内 C++程序员的能力的想法是不现实的。

也许正是因为“术业有专攻”，对 vckbase 来说，是一次前所未有的机遇和挑战。如何有成效地快速提升国内 C++程序员的编程水平？ vckbase 决定在网站上开辟“视频教程-大讲堂”频道，以后广大网友可以通过视频来系统地学习各种与 C++编程技术课程。这不得不说是 C++程序员们的一大福音！

vckbase 找到了国内几十个候选主讲人，经过筛选最后选中了 UIPower 的创始人阚海忠先生。在与阚海忠先生谈起视频大讲坛想法后，他给筹办大讲堂的初衷表示认同，并提出了一些建议。对于 vckbase 来说，拍摄视频教程是第一次接触，所以在前期的录像选址、课堂设置、视频后期制作等都作了一番尝试。一切准备好后，开始了紧张的拍摄。

自视频发布以来，vckbase 收到了很多网友热情洋溢的来信，信中对视频大讲堂的做法大加赞赏，同时也给 vckbase 提出了很多非常宝贵的建议，比如视频背景可以搞的更加专业些，可以加上字幕让不方便在上班时间听声音的网友也可以方便观看视频，再比如课程的内容可以有浅显易懂的也有深入剖析的等等。总之，这些让 vckbase 人感受到了继续做好视频大讲堂的动力。

Vckbase 的成长离不开广大网友的支持，在 2013 年将更加地奋发图强，将 vckbase 推向一个前所未有的新的高度，对此，vckbase 人充满了决心和信心！

视频大讲堂的地址：http://www.vckbase.com/index.php/video

实用第一 智慧密集

书名：Android 4 高级编程（第 3 版）
ISBN：978-7-302-31558-2
定价：98.00 元
作者：（美）迈耶（Meier, R.）著

本书由 Android 权威专家编写，是畅销书《Android 2 高级编程（第 2 版）》的升级图书，涵盖了所有最新的内容，是学习使用 Android 4 SDK 开发移动应用程序的理想指南。本书见解深刻，帮助经验丰富的 Android 开发人员充分挖掘 Android 4 的新特性的潜力，同时讲解了 Android 开发的基础知识，使初学者也可以借助本书入门。作为一本以实用性为目的的指导图书，本书带领您逐步完成复杂程度越来越高的 Android 项目，每个项目中都引入一种新的 Android 平台特性，并着重指出有助于编写引人入胜的应用程序的技术和极佳实践。

本书不仅深入分析了 Android 应用程序的组件和生命周期，还探讨了 Android 的 UI 原理、设计理念和 UI API，使用户界面在手机、平板电脑和电视上都引人注目。

书名：编程语言原理（第 10 版）
ISBN：978-7-302-31112-6
定价：98.00 元
作者：（美）塞巴斯塔（Sebesta, R. W.）著

本书作者 Robert W. Sebesta 是科罗拉多大学斯普林斯分校的计算机科学系的荣誉退休副教授，他从事计算机科学的教学已超过 38 年。

本书从为什么学习程序设计语言入手，深入细致地讲解了命令式语言的主要结构及其设计与实现，内容涉及变量、数据类型、表达式和赋值语句、控制语句、子程序、数据抽象机制、对面向对象程序设计的支持、并发、异常处理和事件处理等方面。最后两章介绍了函数式程序设计语言和逻辑程序设计语言。

本书特色：介绍了编程语言的相关主题；介绍了高级面向对象主题和语言；提供了与一些著名计算机科学家和语言之父的访谈；提供了大量重要的历史史料；深入讨论了几种常见语言的设计问题；提供了函数式和逻辑两种编程泛型书。

书名：iOS 游戏开发入门经典
ISBN：978-7-302-31637-4
定价：59.80 元
作者：（美）阿莱西（Alessi, P.）著

本书首先介绍 Xcode 和 Interface Builder 等必需的工具，然后讲述用于 iOS 开发的 C 和 Objective-C 语言，讨论 Cocoa Foundation 框架和 MVC 体系结构的用法。在介绍基础知识后，本书接着指导您使用库来添加图形、动画和声音，并控制用户交互甚至开发网络对战游戏。几乎每章都列举一个完整实用、简明易懂的游戏示例；为使您确切理解每个步骤的原理，作者 Patrick Alessi 逐行解释所有代码，并最终在章节末尾建成一个完整游戏。

本书特色：指导您创建真正可运行的游戏，带您领略游戏编写之道；详解创建 iOS 游戏的重要库：图形、用户交互、动画和声音；讲述如何使用苹果的框架简化游戏开发；指导您高效地调试和测试游戏。

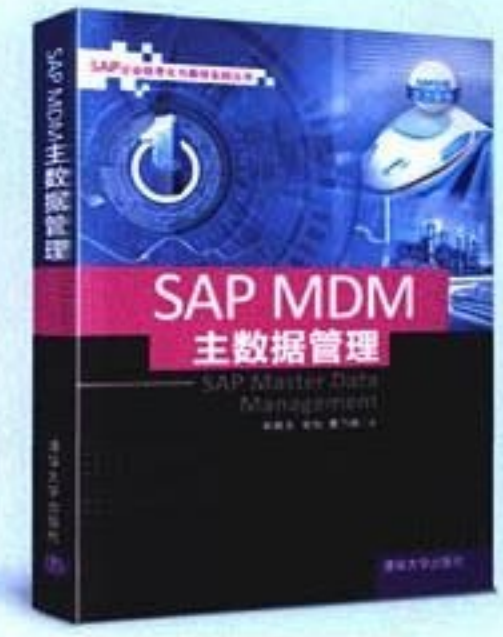
书名：Scrum 实战——敏捷软件项目管理与开发
ISBN：978-7-302-31463-9
定价：39 元
作者：（美）帕姆（Pham, A.），（美）帕姆（Pham, P-V）著

本书为软件项目团队提供了如何成功实施敏捷软件框架 Scrum 的实用指南。本书叙事清晰准确，是第一本由实践者编写的针对现实状况的实用指南。书中描述了如何使项目团队价值最大化，弥补了许多 Scrum 和项目管理书籍缺少的部分，包括如何使用财务术语与高层管理人员交流、如何使用客观的评估技术、软件架构如何适应 Scrum 等。附录提供了案例研究，描述了如何利用本书提到的技术和建议来成功地构建和部署两个软件产品。

本书由畅销书 Managing Agile Projects 的作者 SanjivAugustine 和 Head First Software Development 的作者 DanPilone 作序推荐，为项目团队解决现实问题提供可操作性信息的实用工作指南，书中的具体解决方案可用于各类软件项目。

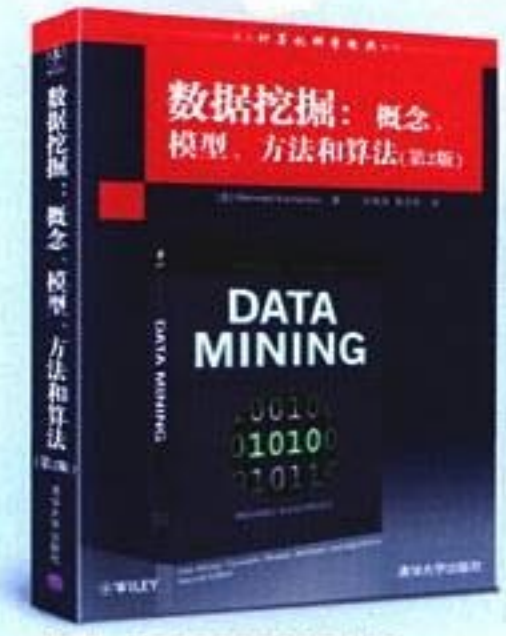
HOLD住你的数据管理

书号：9787302292562
定价：49.80元

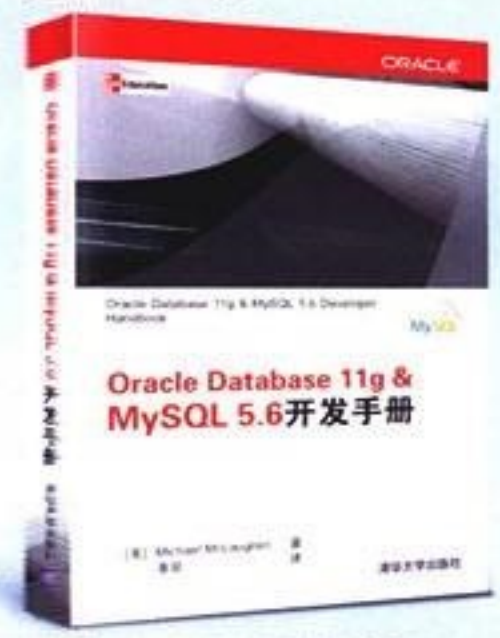


书号：9787302317876
定价：98.00元

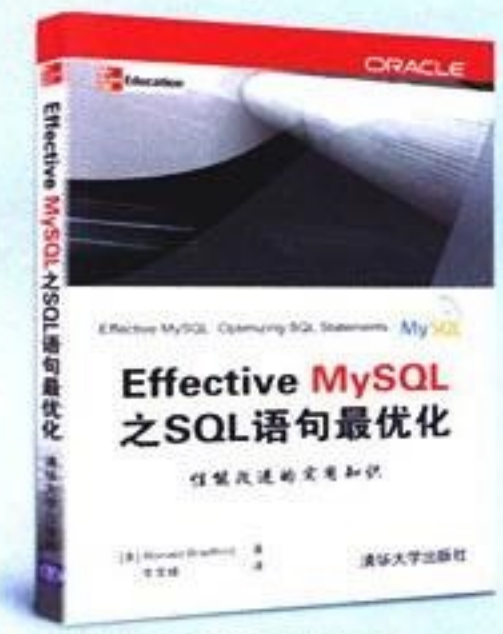
书号：9787302310143
定价：79.80元



书号：9787302307143
定价：59.00元

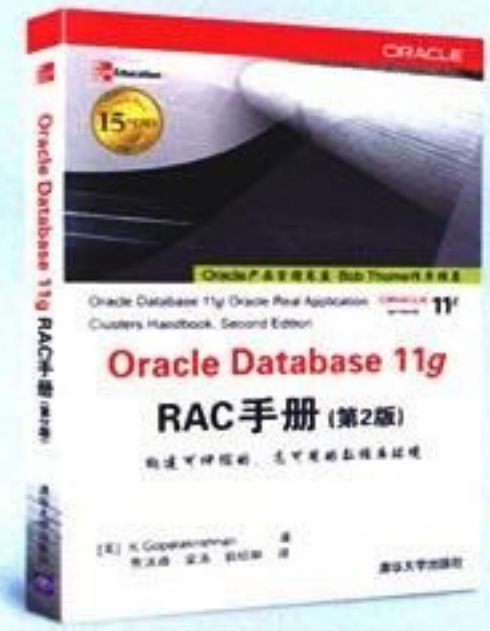


书号：9787302310310
定价：79.80元



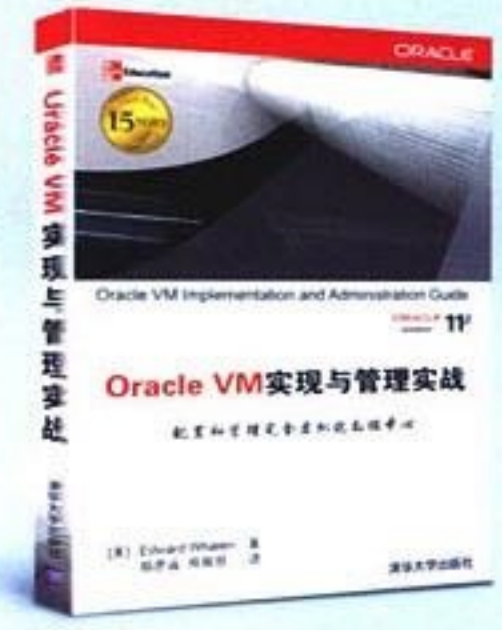
书号：9787302304296
定价：29.00元

书号：9787302304258
定价：50.00元



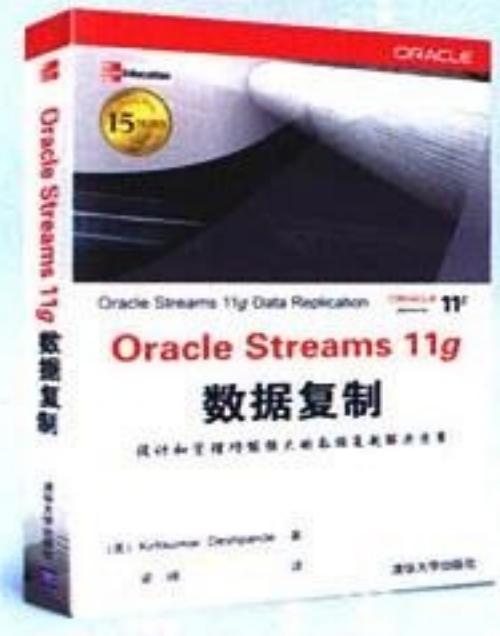
书号：9787302288091
定价：69.00元

书号：9787302286066
定价：59.00元



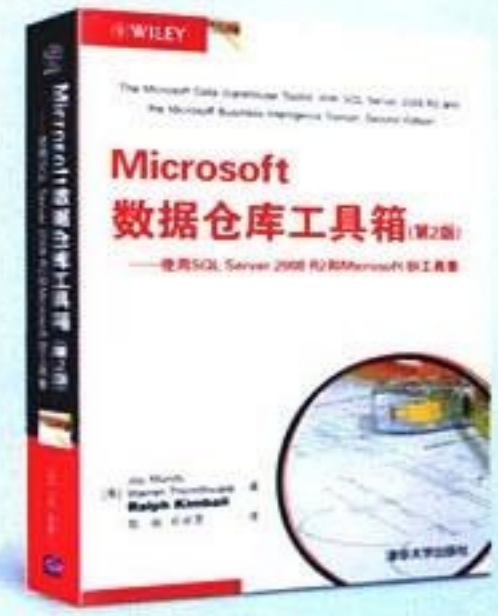
书号：9787302282136
定价：50.00元

书号：9787302282112
定价：59.00元

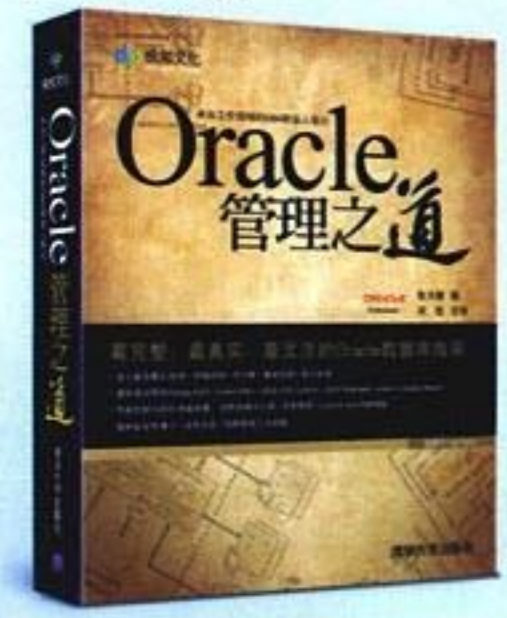


书号：9787302279686
定价：68.00元

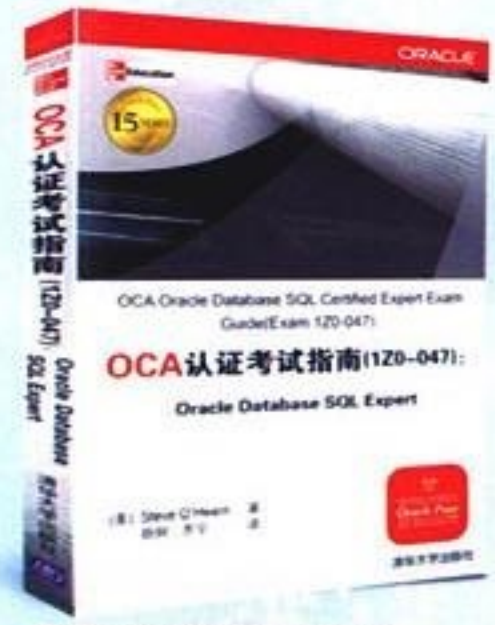
书号：9787302291527
定价：59.00元



书号：9787302283362
定价：78.00元



书号：9787302285403
定价：98 .00元



书号：9787302275398
定价：69.00元

清華大學出版社
TSINGHUA UNIVERSITY PRESS

msup® | Make Professional Discovery mpd

北京 2013.06.15-16　上海 2013.06.22-23　深圳 2013.09.07-08

官网：www.mpd.so　预定购票：400 1818 020

每个微型课程 3 小时实践精华

知名一线研发团队带头人面对面多维度技术体验和经验分享

	I'M Product Manager	I'M Team Leader	I'M Architect	I'M Program Manager	I'M Test Manager
	产品创新/用户体验	团队管理/组织发展	架构设计/技术战略	开发管理/流程再造	测试管理/质量平台
思想 09:00-12:00	产品之美：用户体验设计 徐 锋 麦思博（msup）有限公司资深顾问，畅销书籍《软件需求最佳实践》作者	从技术大脑转向团队领导人大脑 段文韬 曾任职Google中国测试团队经理。2012MPD工作坊成都站“TOP ONE”获得者	雅虎广告定制系统和实验平台分析 邵 峰 曾任美国Yahoo广告产品的架构师	CodeKata:面向对象的JavaScript 姜志辉 IBM中国杰出讲师。2012MPD工作坊厦门站“TOP ONE”获得者	闪电发布，在线测试 袁孟轲 曾任微软美国总部软件开发与测试主管、百度测试总监
过程 13:00-16:00	用户体验的兼容 朱 宏 微软中国用户体验布道师	从知到行--敏捷项目管理的道与术 杨锋镝 麦思博（msup）有限公司资深敏捷咨询顾问	架构的实战过程 周爱民 曾任支付宝公司架构师，有十余年的软件开发、项目管理、团队建设的经验	Phabricator, how the open source project help facebook improve code quality Jason (Jun) Ge Facebook Credits创始开发工程师	实践并解析性能测试及性能分析过程 Zee Gao 麦思博（msup）有限公司资深顾问
榜样 16:15-17:15	微创新：从用户角度发现创新的B面 金错刀 微创新提出者。知名科技商业观察家，爱乐活副总裁	Software Engineering in Practice 刘可乐 曾任Adobe研发经理	云计算实战：Hadoop开发全程代码实战 王家林 曾任三星、华为等公司移动项目平台架构师	敏捷开发实战经验 李勤飞 网易有道研发经理	让Web UI测试跟上开发的速度 殷 坤 东软集团基础软件事业部测试中心主任
管理 09:00-12:00	乐趣项目 黄 冬 土豆网产品技术副总裁	项目经理领导力是怎样炼成的 David.Zhang 曾任IBM(中国)有限公司管理部总监	软件调试案例集锦 张银奎 业界著名系统调试专家	创新项目的敏捷实践 张绍鹏 百度项目管理部架构师，敏捷教练	Lessons Learned in Testability Scott Google Software Design Engineer
实验室 13:00-16:00	基于用户体验的产品创新 林 敏 曾任三星（中国）设计研究所高级经理	从技术走向管理：研发团队与工作管理 陈 皓 曾任亚马逊中国研发经理	AWS development using S3, DynamoDB, and EC2 John Guo an engineer lead at Amazon	实施敏捷产品线方法 徐 昊 ThoughtWorks中国Head of Technology	互联网测试模式和探索式测试实践 高 翔 淘宝软件有限公司资深测试工程师

* 本日程为指导日程，最终日程以大会现场为准，或登陆www.mpd.so查看最新日程。

角色论
大会按照研发角色命名五大分会场
并以岗位角色为中心进行课程规划设计

专注软件研发中心成长
大会按照workshop公开课组织设计
软件开发团队必修课

SPEAKER FROM
国际一线研发团队“教练”
60位知名软件企业一线研发带头人
以工作场景著称的工作坊教学模式

教练—探究团队管理良方
180分钟微课堂
探究团队管理良方

亚太软件研发团队管理峰会（Make Professional Discovery：mpd大会）专注于软件研发中心的快速成长，服务于软件开发团队的技能提升。软件工程的实际应用和软件品质的创新与超越。强调人员、技术、流程和管理的有机结合，注重个体的技能提升与职业发展，研发团队的管理与协作。分享优秀软件研发团队管理实践，这正是mpd的精髓所在!